i2 Analyze

Welcome to the i2® Analyze documentation, where you can learn how to install, deploy, configure, and upgrade i2 Analyze.

i2 Analyze is an integrated set of secure services and stores that provide search, analysis, and storage of intelligence data to authorized users. It presents these functions through the web and desktop applications that connect to it.

Support
The i2 Analyze support page contains links to the release notes and support articles.
i2 Analyze support

Understanding i2 Analyze

i2 Analyze is an integrated set of secure services and stores that provide search, analysis, and storage of intelligence data to authorized users. It presents these functions through the web and desktop applications that connect to it.

When deployed with all components active, i2 Analyze provides key features to its users:

- A structured store for intelligence data with services that enable bulk and targeted create, retrieve, update, and delete operations.
- An extensible framework for connecting to and retrieving intelligence data from sources external to the platform.
- A searchable store for Analyst's Notebook charts that analysts can use for remote storage, and for sharing work with their peers.
- A pervasive data model that is optimized for exploring and visualizing the relationships between records.
- A security model and architecture that together ensure the security of data in motion and at rest.

i2 Analyze enables these features through a range of technologies:

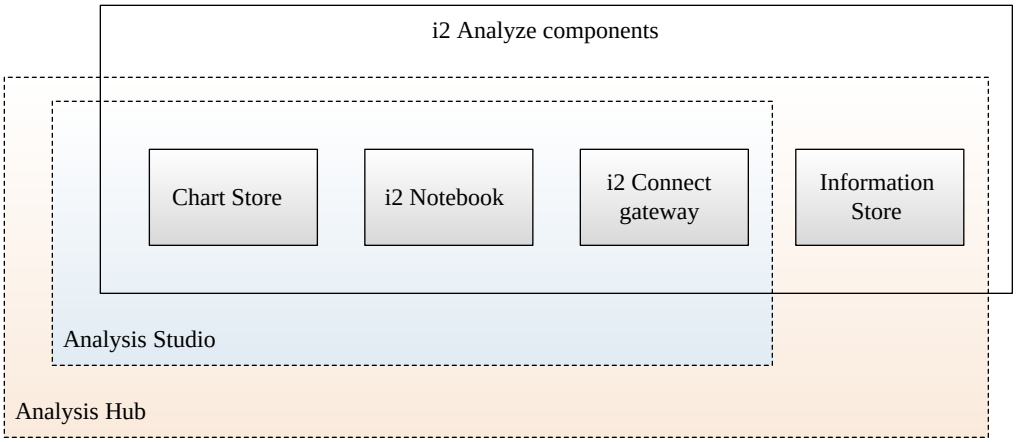
- The i2 Analyze services are deployed in an application on an Open Liberty application server.
- Searching and returning results from intelligence data uses the Apache Solr search index platform.
- The stores for intelligence data and Analyst's Notebook charts are created in a database management system.

The precise feature set of a particular deployment of i2 Analyze depends on the components that you decide to include, which depends in turn on the i2 software that you use it with. The different components of i2 Analyze are subject to licensing agreements.

Components

The data and security models that i2 Analyze provides are a constant in all deployments. Its other features are due to components that you can choose to deploy according to your requirements and the terms of your license.

i2 Group provides i2 Analyze as a part of other offerings including Analysis Studio and Analysis Hub. These offerings entitle you to deploy different components of i2 Analyze.



- For any of the offerings, you can deploy i2 Analyze with the Chart Store, so that users can store and share their charts securely.
- You can also deploy i2 Analyze with the i2 Connect gateway, which allows you to develop and deploy connectors to data sources outside your organization.
- For Analysis Hub, you can also deploy the Information Store, in which users can retain, process, and analyze intelligence data.

Chart Store

The Chart Store has two purposes in a deployment of i2 Analyze. First, it provides Analyst's Notebook users with server-based, secure storage for their charts. Second, in a deployment that does not also include the Information Store, it supports the operation of the i2 Notebook web client.

Users of Analyst's Notebook can upload charts to a Chart Store as easily as they can save them to their workstations. When they do so, they can choose to share charts with their colleagues, or take advantage of i2 Analyze's features for organizing, indexing, and securing charts:

- i2 Analyze authenticates users at login and then uses authorization to control access to all stored charts
- Charts in the Chart Store are subject to the i2 Analyze security model, enabling per-team or per-user access
- i2 Analyze indexes both the contents and the metadata of uploaded Analyst's Notebook charts, so that users can search and filter large numbers of charts quickly and easily

  Note: For each chart in the store, the generated index contains information from i2 Analyze records and Analyst's Notebook chart items. i2 Analyze does not index the contents of any documents that are embedded in a chart.
- The chart metadata itself is configurable, so that the options for categorizing and searching charts reflect the users' domain.

Users of the i2 Notebook web client can search for and visualize data from an Information Store and any external sources that are connected through the i2 Connect gateway. However, not all deployments

of i2 Analyze include both of those components. In a deployment that does not have an Information Store, the i2 Notebook web client requires the Chart Store, which provides short-term storage for the web charts that users create.

For information on deploying i2 Analyze with the Chart Store, see the deployment documentation.

i2 Connect gateway

When you deploy i2 Analyze with the i2 Connect gateway, you gain the option to develop or purchase connectors that can acquire intelligence data from external sources. When a user of Analyst's Notebook or the i2 Notebook web client makes a request to an external data source, the connector converts results from their original format into the entities, links, and properties of i2 Analyze records.

When i2 Analyze accesses data through a connector, the source is not modified. After the data is vetted and verified, and provided that the new or modified entities and links have compatible types, Analyst's Notebook users can upload the records to an Information Store.

For each external data source, an organization can choose between setting up a connector to access the data in the external location, and ingesting the data into an Information Store. The choice depends on a number of factors, including the following considerations:

- **Availability**

  A connector is typically useful where a data source is constantly available. If a data source is not always available, ingestion to the Information Store provides users with constant access to a snapshot of the data.

- **Currency**

  If an external data source is updated very frequently, and it is desirable to have the most current data returned at each request, a connector can be used to achieve this. If regular refreshes are appropriate for the data, but moment-to-moment currency is not required, the data source could instead be ingested to the Information Store on a regular schedule.

- **Quantity**

  The Information Store is designed for high-volume data. However, an organization might not want to create a duplicate copy of a very large external data source that can be made available using a connector.

- **Terms of use**

  For reasons of security, privacy, or commercial interest, a third-party provider of data might not permit the organization to store a copy of the data source in their own environment. In this situation, a connector must be used to access the external data source.

For more information about how the i2 Connect gateway and connectors provide access to external data, see Connecting to external data sources. For examples of creating and deploying connectors, see the open-source project at https://github.com/i2group/Analyze-Connect.

Information Store

The Information Store is a secure, structured store for intelligence data that analysts and other users can search and visualize. i2 Analyze provides services that enable bulk and targeted create, retrieve, update, and delete operations on Information Store data. In deployments that include it, the Information Store also fulfills the functions of the Chart Store.

Records in the Information Store can be loaded automatically from data sources external to i2 Analyze, or they can be imported or created by individual Analyst's Notebook users. To cope with the different

requirements of bulk and targeted operations, the store distinguishes between system- and analyst-governed data:

- System-governed data is loaded into the Information Store in bulk. i2 Analyze includes a toolkit that accelerates the process of extracting and transforming data from external sources and loading it into the Information Store. You can arrange for correlation to take place during this process, and for subsequent loads from the same source to update or delete data that the Information Store has seen before.
- Analyst-governed data is added to the Information Store through Analyst's Notebook, when users upload records that they've imported or created directly on their charts. To optimize that process, i2 Analyze provides matching functionality, so that users are alerted when records that they create match records that are already in the Information Store.

Subject to authorization, Analyst's Notebook users can modify analyst-governed records in the Information Store, but system-governed records are read-only. Apart from that distinction, all records in the Information Store can searched for, compared, and analyzed in exactly the same way, regardless of governance.

Users of the i2 Investigate and i2 Notebook web clients cannot modify any records in the Information Store, but they can search for and visualize records from the store using the tools that those applications provide.

To understand the structure of data in the Information Store, see The i2 Analyze data model. To learn how to deploy i2 Analyze with the Information Store, see Creating an example with an Information Store and Creating a production deployment. For more information about using the ETL toolkit to populate the Information Store with system-governed data, see Ingesting data into the Information Store.

Data model

The i2 Analyze data model governs the structure of the data that i2 Analyze processes. That structure, consisting of entities and links and the properties they contain, is fundamental to the analytical and presentational functionality provided by the offerings that i2 Analyze is part of.

The i2 Analyze data model

The i2 Analyze data model is the foundation on which the abilities of the platform to manage, process, and visualize data are based. On the server and in the user interface, i2 software models data in terms of entities, links, and properties (ELP).

- An *entity* represents a real-world object, such as a person or a car.
- A *link* associates two entities with each other. For example, a Person entity might be associated with a Car entity through an Owns link.
- A *property* stores a value that characterizes an entity or a link. For example, a Person entity might have properties that store their given name, their surname, their date of birth, and their hair color.

Every deployment of i2 Analyze contains descriptions of the kinds of entities and links that can appear in the data for that deployment. These descriptions also state exactly what properties the entities and links can have, and define the relationships that can exist between entities and links of different kinds.

In i2 Analyze, *schemas* provide these deployment-specific descriptions of the data model:

- If your deployment includes an Information Store, a schema determines both its structure and the shape that data must have in order for the Information Store to process it.

- If your deployment includes the i2 Connect gateway, each connector can have a schema that describes the shape of the data that it provides. Additionally, the gateway can have one or more schemas that describe the data for several related connectors.

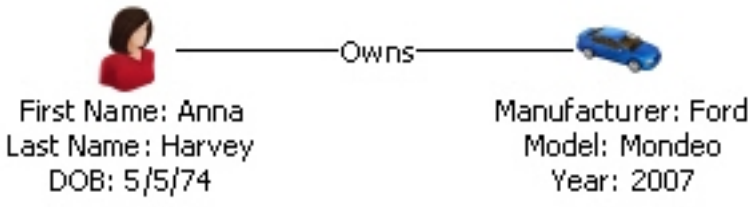
To create a schema for an i2 Analyze deployment, you must examine the data that is likely to be available for analysis, and understand how that data is used during an investigation. Understanding the aims of your investigations is key to helping you to organize data effectively.

ELP relationships in i2 Analyze

Depending on the nature of the data that you want to process in your deployment of i2 Analyze, you might need to shape it to the ELP (entity, link, property) format. Putting data into ELP format enables many of the analytical functions that i2 Analyze provides.

The simplest ELP relationship involves two entities that are connected with a single link. These kinds of relationships are the building blocks for networks that contain groups and chains of entities with any number of links between them.

In i2 Analyze, a simple relationship that involves two entities, a link, and their properties can be visualized like this example:



Note: Because of the way that these relationships appear in visualizations, the structure is sometimes called a *dumbbell*.

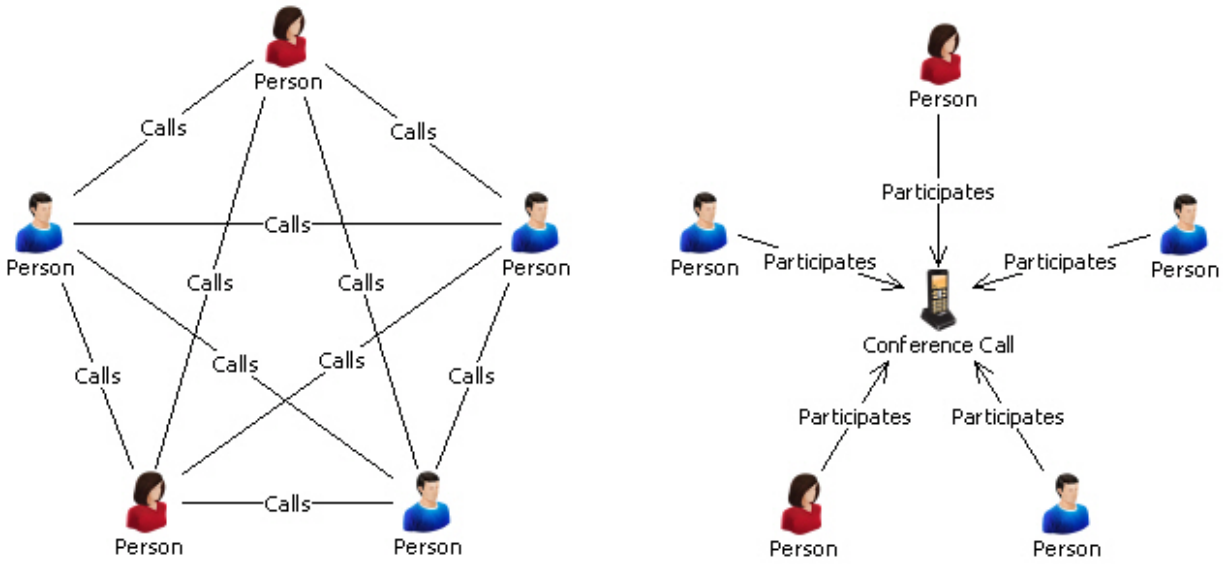
Some of the information that users see in a relationship like this one comes from the data itself:

- For the entity on the left, the data includes the property values "Anna", "Harvey", and "5/5/74".
- Similarly, for the entity on the right, the property values include "Ford", "Mondeo", and "2007".
- The data for the link includes a way of identifying the two entities that it connects.

The remainder of the information in the example comes from definitions in an i2 Analyze schema:

- The default icons for the entities, and the names ("First Name", "Manufacturer") and logical types of their properties, are all defined in an i2 Analyze schema.
- The default label for the link ("Owns") is also defined in an i2 Analyze schema.

In practice, it can be best to make links lightweight and use intermediate entities to model the details of complex associations. Among other things, this approach allows improved modeling of multi-way associations, such as a conference call that has multiple participants. The following diagram shows the difference:



To align your data with an i2 Analyze schema, you must resolve it into the component parts of ELP relationships. If your source is a relational database, it is possible that your tables each correspond to particular kinds of entities, while foreign key relationships form the basis for particular kinds of links. If your source is a delimited text file, it is possible that rows in the file contain the data for one or more entities and links.

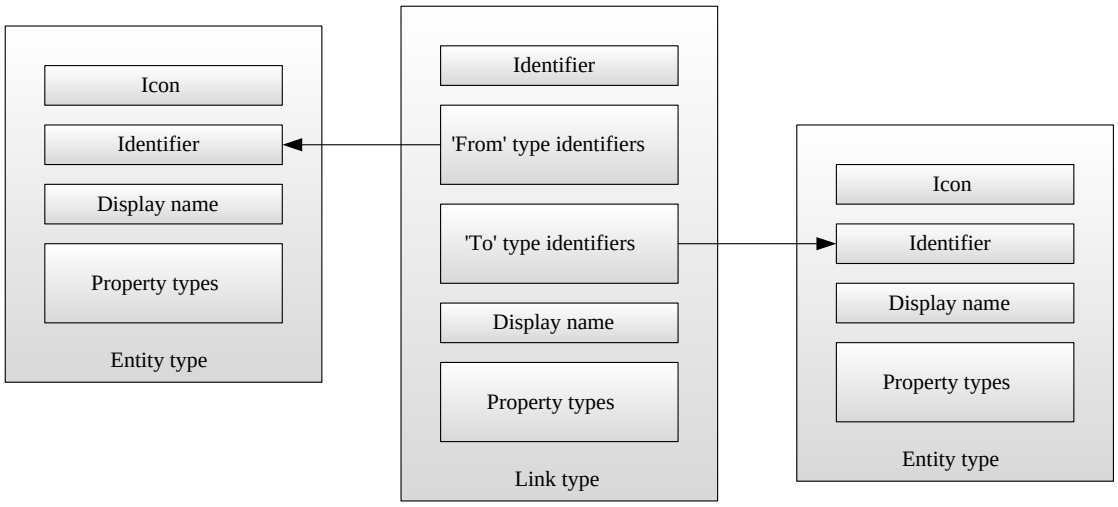
Entity types, link types, and property types

In the i2 Analyze data model, *entity types* and *link types* that determine what entities and links can appear in the data to which the schema applies. The *property types* of those entity and link types determine what properties the entities and links can have.

Entity types and link types

In an i2 Analyze schema, entity types and link types have similar definitions. Among several common features, entity types and link types both have:

- Identifiers, so that types can refer to each other in rules about their relationships
- Names that users see when they interact with entities and links of particular types
- Definitions of the properties that entities and links of particular types can contain

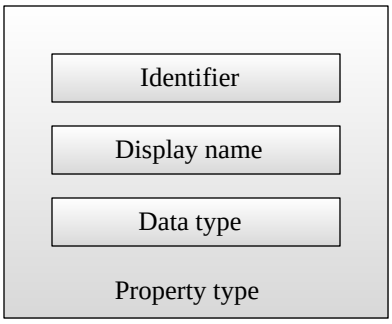


As well as the common features, each *entity* type contains the icon that represents entities with that type in visualizations. *Link* types do not contain icons, but they do contain lists of 'from' and 'to' entity type identifiers. For a link that has a particular link type, these lists determine what entity types the entities at each of the link can have. For example, a link of type Calls might be valid in both directions between two entities of type Person. An Owns link might be valid between a Person and a Car, but would not be allowed in the opposite direction.

For an i2 Analyze schema to be valid, any identifiers that appear in the 'from' and 'to' lists of link types must also appear as the identifiers of entity types.

Property types

In an i2 Analyze schema, entity types and link types both contain property types. For an entity or link record that has a particular type, the property types specify the names and the logical types of the properties that the entity or link can have.



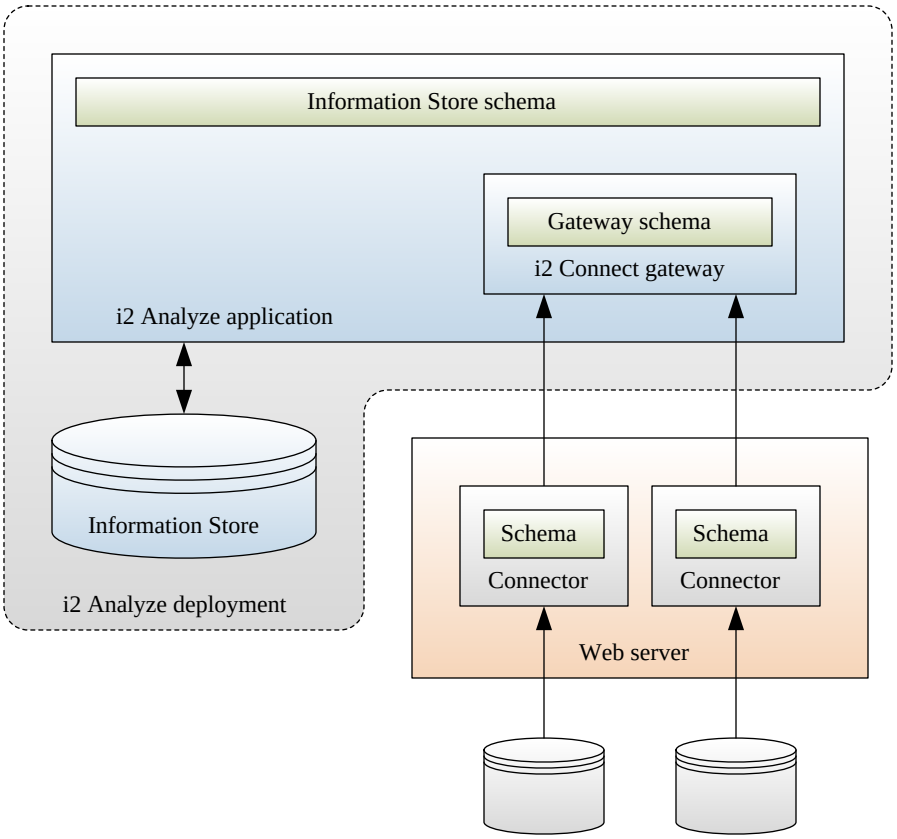
Note: This representation is simplified. A property type can also specify a list of possible values for properties of that type. Furthermore, it can declare whether a property of that type is mandatory for an entity or a link that has the containing type.

i2 Analyze schemas

An i2 Analyze schema is a statement about the categories of data that a deployment can work with, and about the relationships that can exist between data in different categories. As a result, i2 Analyze schemas are responsible for what data looks like when it is visualized; for the types of analysis that can users can perform; and for the structures in which data is stored for retrieval.

Kinds of schema

All deployments of i2 Analyze must contain at least one schema. Some deployments can contain several schemas. All schemas perform the same role, but which kinds of schema you need depends partly on how you want your deployment to behave, and partly on which components of i2 Analyze it uses.



Information Store schema

If your i2 Analyze deployment includes an Information Store, then it requires an Information Store schema. If it includes *only* an Information Store, then the Information Store schema is the only one you need.

i2 Analyze uses the Information Store schema to create the Information Store database. When you later load data into the Information Store, you must shape the data so that it matches the entity and link types in the schema. And if a client application wants to import data with the aim of uploading it to the Information Store, then it uses the schema to ensure compatibility.

Important: Because of its relationship to the structure of the database, an Information Store schema is the most difficult to modify in a production deployment. Only additive changes to the schema are permitted.

Gateway schema

Gateway schemas are relevant only to deployments of i2 Analyze that include the i2 Connect gateway. When the gateway is present, gateway schemas are optional.

The data that connectors return from external sources must have the same ELP structure as other data in i2 Analyze. Sometimes, you can make data from a connector use the types that an Information Store schema defines. Sometimes, a connector returns data with unique types, and defines a schema of its

own. But sometimes, you might have several connectors whose data shares a set of common types. In this situation, you can create a gateway schema and arrange for all the connectors to use the types that the gateway schema defines.

Note: Because gateway schemas are not tied to the structure of data storage, they are relatively easy to modify and redeploy. Developing and testing a gateway schema can be a convenient way to create a schema that you plan eventually to use for the Information Store.

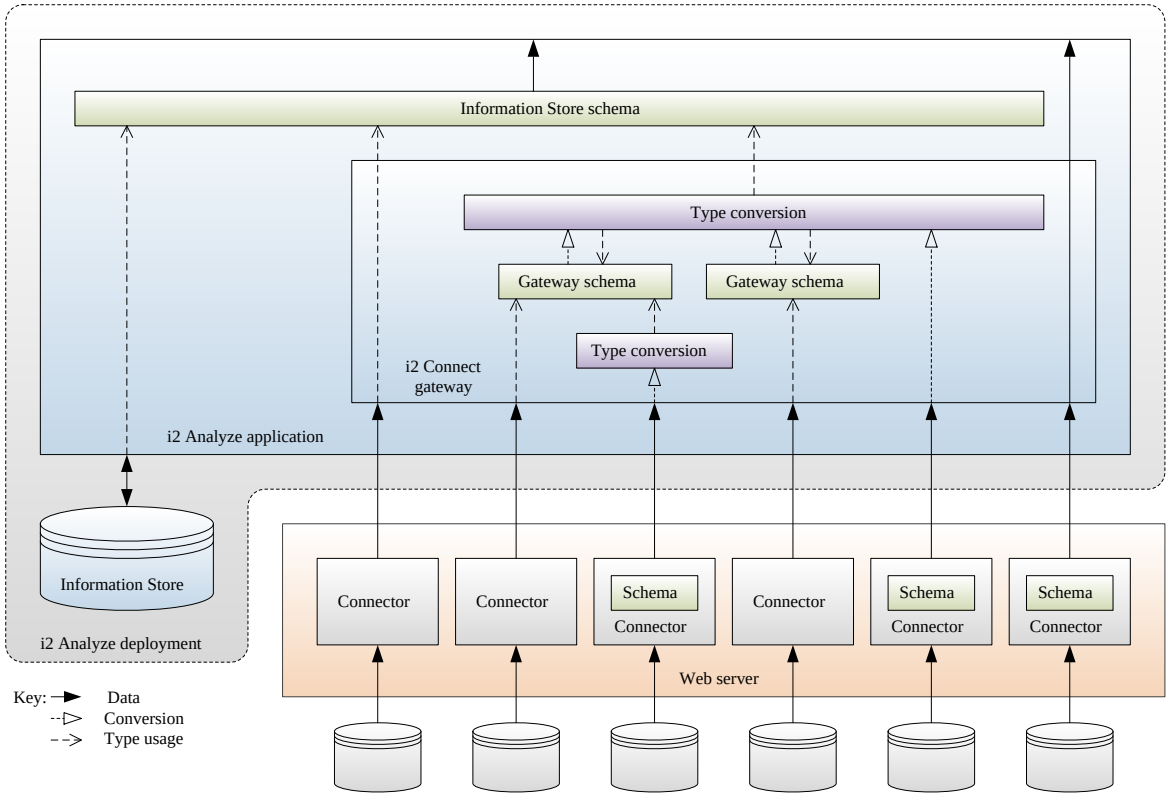
Connector schema

All data in an i2 Analyze deployment must use entity and link types from a schema. If a connector does not or cannot use types from an Information Store schema or a gateway schema, then it must provide its own definitions of the types that it uses.

A connector schema can be appropriate when you are prototyping a connector to a new data source, or when you know that data from a particular source is subject to frequent changes. Alternatively, you might be using or creating a connector that is designed for use in multiple i2 Analyze deployments. In that case, it can be helpful for the connector to come with definitions of its own types.

Schema type conversion

An i2 Analyze deployment that includes the i2 Connect gateway and connectors to external data sources is likely to involve several schemas. Client software can visualize data that uses types from any schema, display its property values, and subject it to structural analysis. However, data can be uploaded to an Information Store only if it uses types from that store's schema. And users can perform comparative analysis only between data that uses the same types.



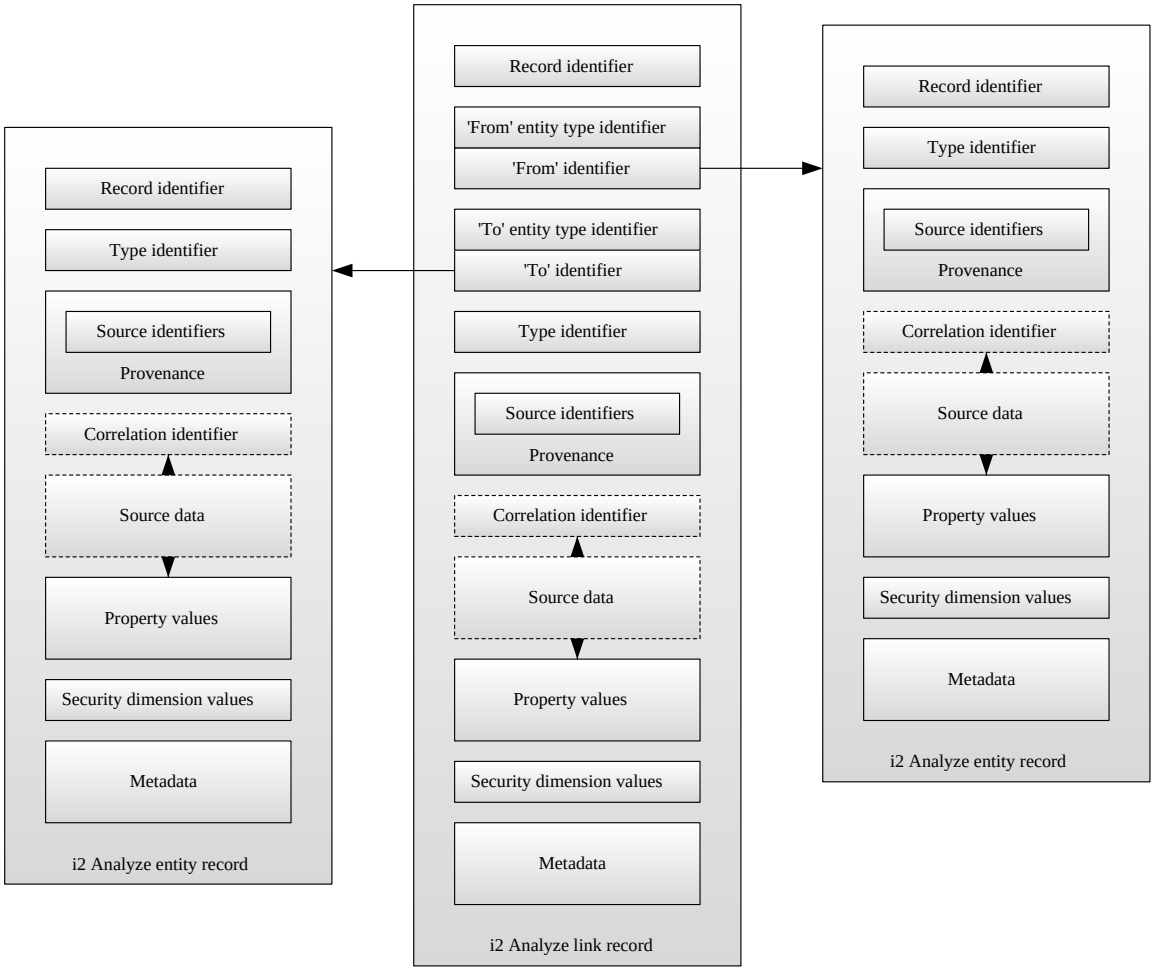
To enable i2 Analyze to treat data from all sources similarly, you can arrange for records to be converted from one type to another as they move through the system. To enable this type conversion, you provide one-to-one mappings between types from different schemas. For example, if you map the types from a connector schema (or a gateway schema) to types in the Information Store schema, then you can upload data from that connector to the Information Store. If you map types from several connector schemas to the types in a gateway schema (or the types in one gateway schema with those in another), it becomes possible to use data from one connector as seeds for searches of another.

Data in i2 Analyze records

i2 Analyze deployments use *i2 Analyze records* to realize the entity and link types that schemas define. i2 Analyze records contain the property data for entities and links, plus metadata that enhances the analysis that users can carry out.

An i2 Analyze schema defines the types that determine what i2 Analyze records can represent. Every i2 Analyze record has a type that an i2 Analyze schema defines. If a record has a link type, then the record represents a link - it is a *link record*. If a record has an entity type, then it is an *entity record*.

This diagram shows how entity and link records compare, and how they are related to each other. It also introduces some other features of the data in i2 Analyze records.



Note: The diagram contains some simplifications:

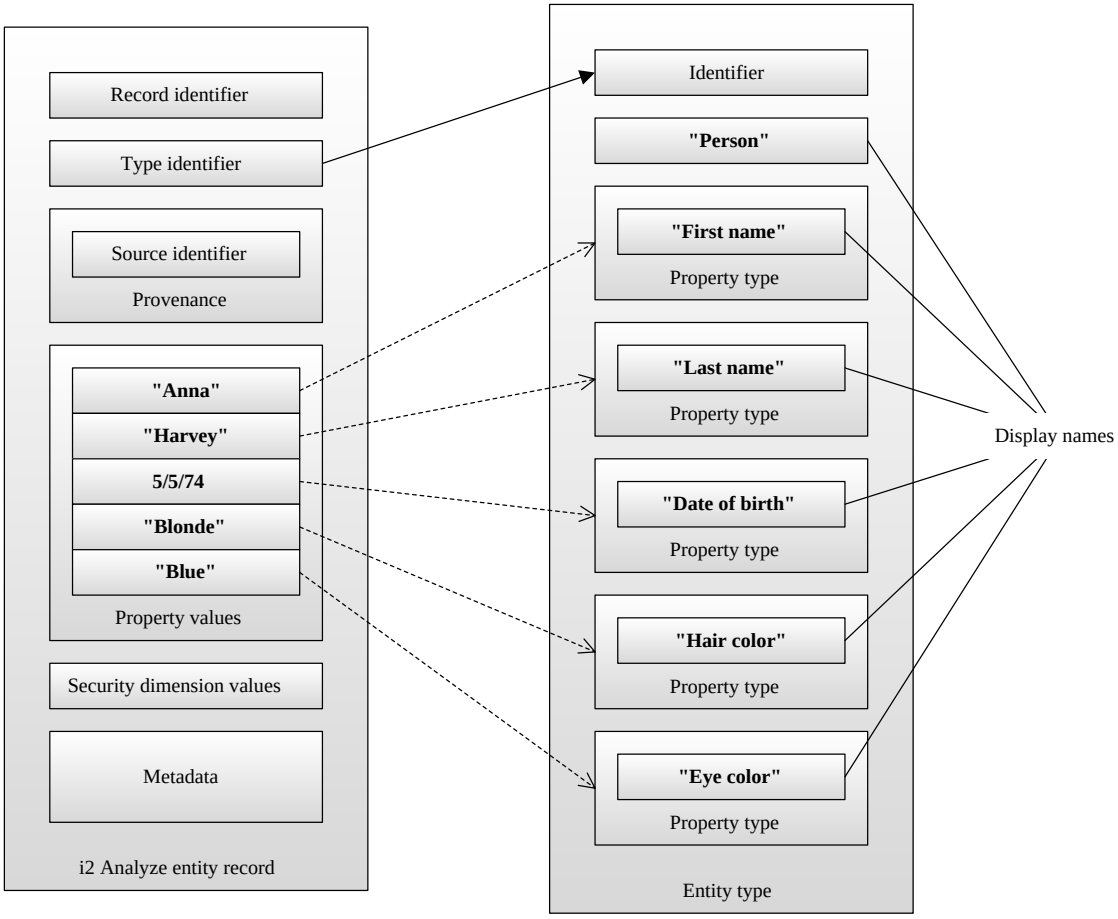
- *Provenance* is about the data sources that contributed to a particular i2 Analyze record. When the property values of an i2 Analyze record represent data from more than one source, that record can have more than one piece of provenance.
- When an i2 Analyze record has more than one piece of provenance, it can contain all the data from all the contributing sources. In that case, the property values that the record presents are derived from the source data.
- *Metadata* includes the following information:
  - Timestamps, which reflect when data in an i2 Analyze record was created or modified
  - *Source references*, which describe the sources that the data in a record came from
  - *Notes*, which users can write to add free-form text commentary to a record

  For link records, the metadata also includes information about the strength and direction of the link.

As an example of how to represent a simple entity that contains data from a single source, consider the following information about a person:

```
Full name:      Anna Harvey
Date of birth:  5/5/74
Hair color:     Blonde
Eye color:      Blue
```

The following diagram shows one way to represent this information as an i2 Analyze record:



Note: An i2 Analyze entity record can contain properties that have any of the property types that the entity type defines. However, one record can contain only one property of each defined type.

The diagram also shows how the property types in the schema only partially determine the contents of an i2 Analyze record. Some of the other contents are due to the security schema, while others still are about identification:

- All i2 Analyze records contain security dimension values, which i2 Analyze uses to determine the access level to the record that a particular user has.

- When they enter the system (through ingestion to the Information Store, or through Analyst's Notebook), i2 Analyze records receive a universally unique *record identifier*. This identifier is permanent for the lifetime of the record. If they have the necessary access level, any user of the system can use the record identifier to refer to a particular record.
- i2 Analyze records that began life in an external data source contain one or more pieces of provenance. Each piece has a source identifier that references the data for the record in its original source. One record can have provenance from more than one source.

  Note: For records in an Information Store that were loaded through ingestion, source identifiers have the additional feature of being unique within the store. These source identifiers are known as *origin identifiers*.
- All i2 Analyze records can contain timestamps in their metadata that specify when source data for the record was created or edited.
- i2 Analyze link records contain an indication of their direction. i2 Analyze considers links to go 'from' one entity 'to' another. The direction of a link can be with or against that flow, or it can run in both directions or none.

When i2 Analyze records are stored in an Information Store, they contain a few extra pieces of data:

- All i2 Analyze records retain timestamps in their metadata for when they were first created or uploaded to the Information Store, for when they were most recently uploaded, and for when they were last updated.
- All i2 Analyze records can contain a *correlation identifier*. If two records have the same correlation identifier, the platform considers that they represent the same the real-world object and might merge them together.

Your data sources are likely to contain some, but not all, of the data that i2 Analyze records require. To enable an Information Store to ingest your data, or to develop a connector for the i2 Connect gateway, or to write an import specification, you must provide the extra information to i2 Analyze.

Identifiers in i2 Analyze records

i2 Analyze records make extensive use of identifiers. Records use identifiers to refer to their type in an i2 Analyze schema, to their original source data, and to other records in ELP relationships. Preparing data for compatibility with i2 Analyze often involves creating or providing the identifiers that form the basis for the reference mechanisms.

Type identifiers

Every i2 Analyze record contains a type identifier, which is a reference to one of the entity types or link types that a schema defines. When you create an ingestion mapping file, an import specification, or a connector, you must arrange for each incoming record to receive the identifier of a type definition.

Every i2 Analyze *link* record contains two further type identifiers, which are references to the entity types of the records at the ends of the link. You must arrange for incoming link records to receive these identifiers as well.

This strong typing of records in i2 Analyze is key to the analytical functions that the platform provides. It allows users to consider not only the existence of relationships between records, but also the nature of those relationships. Schemas define exactly what relationships to allow between record types, and i2 Analyze enforces those rules during record creation.

Record identifiers

i2 Analyze records are created when you ingest data into the Information Store, or when a user creates an item that contains an i2 Analyze record on the chart surface by:

- Importing data through an import specification
- Adding the results of an operation against an external source
- Using an i2 Analyze palette in Analyst's Notebook

At creation, every i2 Analyze record automatically receives a universally unique record identifier that is permanent for the lifetime of that record. Users and administrators of an i2 Analyze deployment can use the record identifier as a convenient way to refer to a record in features such as text search and the Investigate Add-On.

Source identifiers

The role of a source identifier is to refer to the data for a record reproducibly in its original source. If a record represents data from several sources, then it contains several source identifiers. The nature of a source identifier depends on the source and the record creation method, and sometimes on whether the record is a link or an entity.

When you write ingestion mappings or develop connectors for the i2 Connect gateway, you are responsible for providing the identifying information. For example, if the original source is a relational database, then entity data is likely to have ready-made source identifiers: table names and primary keys. Link data can also have ready-made source identifiers, but it might not, especially if the relationship that the link represents exists only as a foreign key.

If the source of a record is a text file, then the file name might form part of the source identifier, along with some reference to the data within the file.

Note: Source identifiers are not displayed to end users, but they are a part of the data that records contain. Avoid including sensitive information such as passwords, or configuration detail such as IP addresses. Assume that any information you use as part of a source identifier might be read by users of the system.

Origin identifiers

In general, source identifiers are not certain to be unique within a deployment of i2 Analyze. Several users might independently retrieve the same data from an external source, resulting in several records with the same source identifier. However, when you *ingest* data into the Information Store, i2 Analyze compares the incoming source identifier with existing records. If it finds a match, i2 Analyze updates a record instead of creating one.

The source identifiers that records receive during ingestion therefore *are* unique within i2 Analyze, and they have a special name in this context. They are called *origin identifiers*.

Correlation identifiers

The purpose of a correlation identifier is to indicate that the data in an i2 Analyze record pertains to a particular real-world object. As a result, correlation identifiers are usually related to property values rather than other identifiers. (For example, two Person records from different sources that contain the same social security number are likely to contain data about the same real person.) When two records have the same correlation identifier, they represent the same real-world object, and are candidates to be merged.

When you ingest data into the Information Store, you can provide a correlation identifier for each incoming record. For more information about correlation identifiers and how to create them, see Correlation identifiers.

Security of i2 Analyze records

i2 Analyze records are subject to the i2 Analyze security rules. The security schema defines the security model for your i2 Analyze deployment, and every i2 Analyze record must have at least one value from each security dimension in the schema.

When a user runs a query, i2 Analyze looks up which groups the user belongs to, and determines their security permissions. Then, it compares their security permissions to the security dimension values of the records in the query results. In this way, i2 Analyze calculates which records the user has access to.

If your deployment of i2 Analyze includes an Information Store that you populate through ingestion, then you must add security information to the data during that process. Each time the process runs, you can specify which security dimension values the incoming records receive:

- If you decide that all the data from a given external source must have the same security settings, you can specify the same dimension values for records of all types.
- Alternatively, you can dictate that all the records of a particular entity type or link type receive the same security dimension values.
- You can also configure the process so that individual records receive security dimension values that you specify or determine programmatically.

Note: In this version of i2 Analyze, you can change the security dimension values of an ingested record only by ingesting it again with a different set of values.

All other i2 Analyze records receive dimension values when users create them in Analyst's Notebook by importing them, or searching an external source, or by using an i2 Analyze palette. These records start with default values that users can edit in the same way that they can edit property values.

Storing charts in i2 Analyze

In addition to storing records in the Information Store, i2 Analyze can provide a shared Chart Store for Analyst's Notebook charts. Users of Analyst's Notebook can search for, download, edit, and upload charts in the i2 Analyze Chart Store.

To store charts, i2 Analyze uses a similar model to the one that it uses for storing records in the Information Store. Charts in the Chart Store benefit from features such as timestamps, notes, and source references. They also use the same security model as the rest of the system, so user access to charts is controlled in the same way as user access to records.

Extensions to the record model mean that the Chart Store indexes the *contents* of a chart for searching, in addition to its notes and properties. It also keeps an image of the chart's appearance when it was uploaded. Furthermore, the Chart Store retains a version history and performs basic version control operations for its contents.

Note: At this version of i2 Analyze, the Chart Store treats charts as stand-alone entities and does not model connections between them.

To learn more about the information in the Chart Store and what tasks an administrator of i2 Analyze can use it for, see Retrieving chart metadata.

Security model

All data in i2 Analyze can be secured so that only the users who are supposed to interact with it are able to do so. Using the i2 Analyze security model, you can decide what access users have to records and features, based on their membership of user groups.

Categorizing users

In i2 Analyze, all users are members of one or more groups. For example, there might be a group of "administrator" users. There might be separate groups of users for each operational team in your organization. There might be a group of users with higher security clearance than others. Or there might be a group of users who need access to a particular piece of functionality.

System groups

System groups are set up by a system administrator. System groups are used to control all aspects of access to records and features in i2 Analyze, including artifact sharing.

Custom groups

Custom groups can be created and managed by any users whose system group membership permits them to do so. At this version of i2 Analyze, custom groups are used only to enable their members to share artifacts with each other.

Categorizing records

Just as users of i2 Analyze are categorized, so too are records, according to a range of deployment-specific criteria. For example, records might be categorized according to the nature of the information they contain, or how sensitive that information is.

Security schema

To make sure that users see only the records that they are allowed to see, every deployment of i2 Analyze has a security schema. The security schema defines the categories into which records must be placed, and the relationships that determine what access users get to records in a particular category.

In other words, the i2 Analyze security schema allows you to create rules that say things like, "Users with low security clearance cannot see sensitive records," or "Users in Group 1 can only see records whose source was signals intelligence." i2 Analyze then combines the rules predictably, on a per-record and per-user basis.

Important: Orthogonal to this security model, i2 Analyze supports blanket controls over the visibility of records with particular types. You can specify that only users in certain system groups can see records of a specific type, and that all records of that type are invisible to all other users, regardless of security schema categories. For more information about this functionality, see Item type security.

i2 Analyze security dimensions

In the i2 Analyze security model, a *security dimension* is a way to categorize a record, with the aim of using its category to determine whether particular users are allowed to view or modify it. The available security dimensions in a deployment of i2 Analyze are specific to that deployment, and they are defined in its *security schema*.

A deployment of i2 Analyze might need several different ways to categorize records:

- Records might be categorized by their security classifications

- Records might be categorized by the type of intelligence that produced them
- Records might be categorized by the operational teams who are allowed to access them

As a result, the deployment requires several security dimensions. Each dimension contains a set of values that records can have in order to classify them within that dimension.

To continue the example, the three dimensions might contain values as follows:

- **Security classification**

  Top Secret, Secret, Confidential, Restricted
- **Intelligence type**

  Human Informant, Open Source
- **Operational team**

  A, B, C

As a result of these definitions, for example, it is possible to mark a record as containing confidential information derived from a human informant, to be available to users in Team B.

Ordered and unordered

In some dimensions (such as security classification), the possible values form a sequence from which each record takes a single value. In these *ordered dimensions*, the values act as levels, where each value supersedes all the values after it. If a record is "Top Secret", it cannot be "Restricted" at the same time.

In dimensions such as operational team, where the values do not form a sequence, records can take one or more values. You can use the values of an *unordered dimension* to say that a record is available to users in Team B or Team C - or, alternatively, to users who are in both Team B *and* Team C.

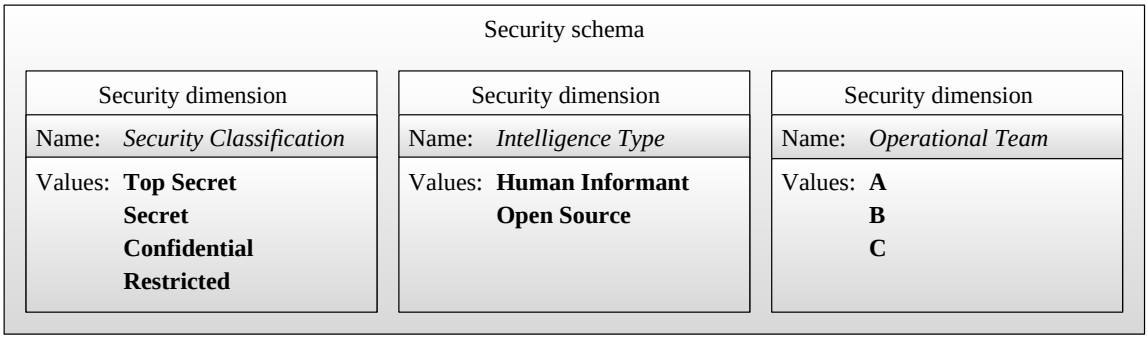
Resolution mode

For an unordered dimension, the *resolution mode* tells i2 Analyze how to behave when a record has more than one value from it. By default, the resolution mode is `ANY`, which means that the record is available to users who qualify according to any of the assigned values ("Team B or Team C").

The other resolution mode is `ALL`, which makes a record with multiple dimension values available only to users who qualify according to all of them ("Team B and Team C").

Rules for records

Every record in an i2 Analyze deployment must have at least one value from each of the security dimensions in that deployment. There is no such thing as an "optional" dimension. For example:



Record X	
Security Classification:	**Restricted**
Intelligence Type:	**Human Informant**
Operational Team:	**C**

Record Y	
Security Classification:	**Secret**
Intelligence Type:	**Open Source**
Operational Team:	**A, B**

There are no restrictions on the numbers of dimensions or values that a security schema can define. Keep in mind, though, that the more dimensions there are, the more complicated it becomes to maintain the security schema. Also, for performance reasons, try to avoid using the ALL resolution mode with a dimension that has more than 100 values.

i2 Analyze security permissions

In i2 Analyze, security permissions provide the link between the security dimension values that a record has, and what users are allowed to do with that record. The platform calculates whether users can see or edit a record according to the permissions of the system user groups to which they belong.

The result of the calculation that i2 Analyze performs is that for any record, a user receives one of three security access levels:

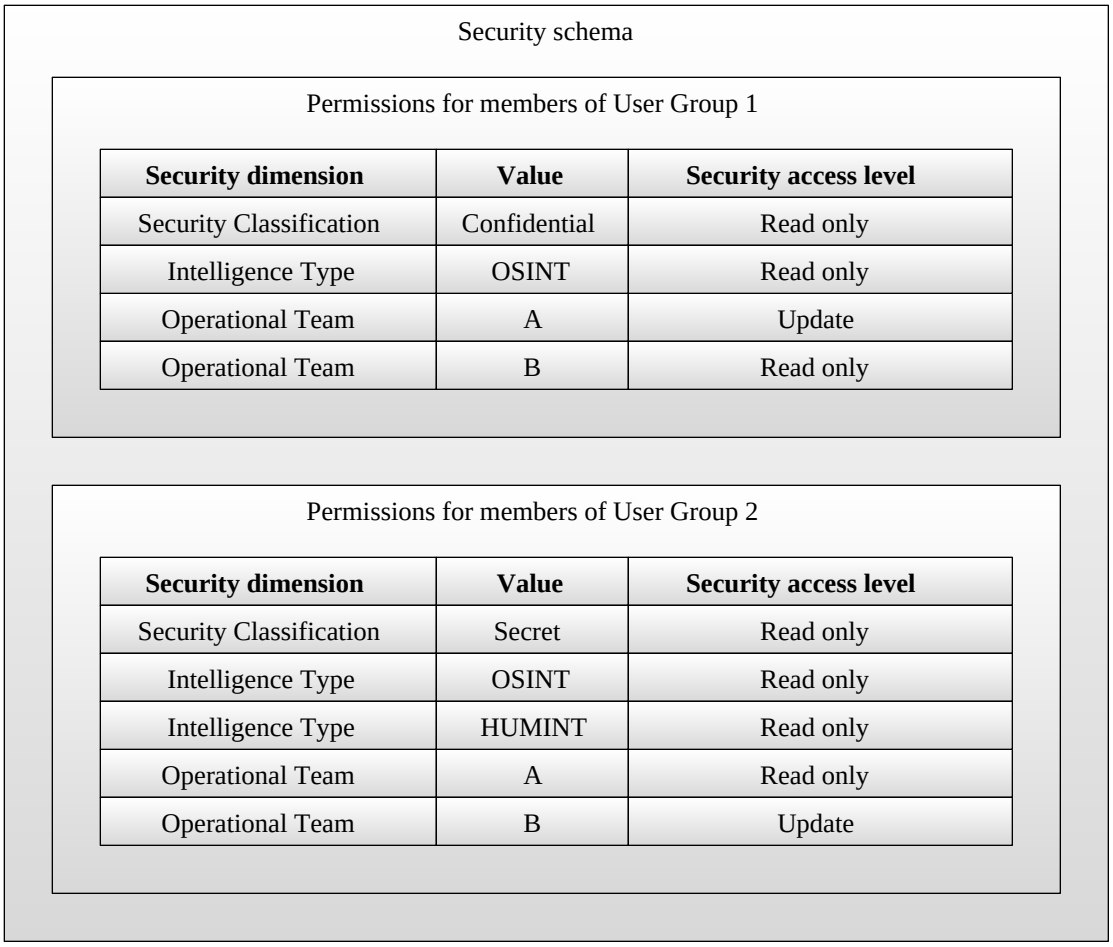
- **None**

  The user has no access to the record. The user cannot examine the record data, or even know that the record exists.
- **Read only**

  The user has read-only access to the record and its data.
- **Update**

  The user can read, modify, and delete the record and its data (subject to their command access control permissions).

In an i2 Analyze security schema, the security permissions for a system user group define mappings from dimension values to access levels. Users receive the security access levels that their groups indicate for the dimension values of a record.



Security schema

Permissions for members of User Group 1

Security dimension	Value	Security access level
Security Classification	Confidential	Read only
Intelligence Type	OSINT	Read only
Operational Team	A	Update
Operational Team	B	Read only

Permissions for members of User Group 2

Security dimension	Value	Security access level
Security Classification	Secret	Read only
Intelligence Type	OSINT	Read only
Intelligence Type	HUMINT	Read only
Operational Team	A	Read only
Operational Team	B	Update

For example, a dimension value might mark a record as containing open source intelligence (OSINT). In the diagram, the permissions for users in Group 1 say that they should have the "Read only" access level on records with that dimension value. However, the access level that users eventually receive depends on how all their permissions combine.

Note: It is not compulsory for a set of permissions for a user group to provide an access level for every value of every dimension. Any dimension value that does not appear in a set of permissions implies the default "None" access level, *unless* the missing value comes later in an *ordered* dimension than a value that does appear.

For example, consider our Security Classification dimension, whose ordered values are Top Secret, Secret, Confidential, Restricted.

If a particular set of permissions associates the "Read only" access level with Restricted records (and makes no other setting), then the default access level for Confidential records is "None". However, if the permissions associate the "Read only" access level with *Confidential* records instead, then users in the same group receive that access level for Restricted records as well.

Combining permissions

In practice, records have several dimension values, and users can be members of several system groups. As a result, users generally receive access levels from more than one permission. i2 Analyze computes a single security access level from all the contributing permissions.

An i2 Analyze system administrator must arrange the security schema so that all users can receive at least the "Read only" access level for at least one value in every dimension. In other words, it must be possible for all users to see at least some records.

Note: If the security schema uses a permissions provider, users can have permissions that are based on their login credentials as well as group membership. This detail does not affect how i2 Analyze processes security permissions.

Security model example

The purpose of the security model is to enable the platform, at any moment, to determine whether a user can see a particular record - and, if they can, whether they can also edit it. The platform performs this calculation according to a consistent set of rules.

At their simplest, when all unordered dimensions have the default ANY resolution mode, the rules for determining access work like this:

- A user can see a record if they receive "Update" or "Read only" access for at least one of the dimension values that the record has from each security dimension.
- A user can edit a record if they receive "Update" access for at least one of the dimension values that the record has from each security dimension.

When an unordered dimension has the ALL resolution mode, the rules for that dimension change so that "at least one" in those descriptions becomes "all".

If a user receives multiple permissions that specify different access levels for the same dimension value, the calculation uses the most permissive level.

Calculation with ANY dimensions

For example, consider the following record, which has one value for each of two security dimensions, and two values for a third.

Record	
Security Classification	**Confidential**
Intelligence Type	**Open Source**
Operational Team	**A**, **B**

Then, consider a user who has the following aggregated security permissions.

Security dimension	**Value**	**Access level**
Security Classification	Secret	Read only
Security Classification	Confidential	Update

Security dimension	Value	Access level
Intelligence Type	Open Source	Update
Intelligence Type	Human Intelligence	Read only
Operational Team	A	Read only
Operational Team	B	Update

To calculate this user's access to the record, i2 Analyze uses the permissions to determine the access level for each assigned dimension value, and then applies the rules.

Assigned dimension value	Access level
Security Classification: **Confidential**	Update
Intelligence Type: **Open Source**	Update
Operational Team: **A**	Read only
Operational Team: **B**	Update

The user has "Update" access for at least one of the values that the record has in each dimension, and therefore receives "Update" access to the record itself.

If the record had the Secret security classification, the user would not have "Update" access for all values, but would still have at least "Read only". They would be able to see the record.

If the record had the Top Secret classification, the user would have no access according to that dimension value. They would not be able to see the record at all.

Calculation with an ALL dimension

To extend the example, imagine that the Operational Team security dimension has the ALL resolution mode. The list of access levels that the user receives does not change, but the final calculation does.

i2 Analyze now considers *both* of the access levels associated with values from the Operational Team dimension. Because one of those levels is "Read only", that level applies to the dimension as a whole, and therefore also to the record. Our user can still see the record, but they can no longer edit it.

Security architecture

The *authorization* that involves the security schema and permissions and dimensions is one part of the i2 Analyze security architecture. Another part is the *authentication* mechanism through which users are assigned the identifiers and group memberships that enable authorization to take place.

Authentication

At login, the Open Liberty server that hosts i2 Analyze requires clients to authenticate before they can interact with the application. As a result, the range of available authentication mechanisms is determined by the capabilities of Liberty. The requirements on the authentication mechanism are as follows:

- The i2 Analyze application must receive user and group information about a user that are derived from the credentials they present.
- A (potentially) deployment-specific module must be able to map that user and group information onto membership of the system user groups that are named in the security permissions section of the i2 Analyze security schema.

If an authentication system can fulfill these requirements, then it is suitable for use in an i2 Analyze deployment. This documentation describes how to configure authentication that uses Liberty's built-in user registry, Microsoft Active Directory, and claims-based authentication providers including SAML and OpenID Connect.

On successful authentication, the client receives a token. During normal operation, the client passes the token back to the i2 Analyze application, which enforces data access rights with reference to their group memberships.

Authorization

After it receives information about a user and their group memberships from the authentication mechanism, i2 Analyze has an opportunity to augment or modify it (for example, to map from Active Directory group names to system group names) during *provisioning*.

When provisioning is complete, the i2 Analyze application uses the information to determine the access rights of the user to the records that it manages. The security model of i2 Analyze is based on the interaction between the security dimension values that records have, and the security permissions that user groups convey.

i2 Analyze also uses group membership to determine user access to some features of the application. For more information about that aspect of authorization, see Controlling access to features.

Saving and sharing

i2 Analyze users often create *artifacts* that they want to store for repeated use. For example, they might define a structured search that they want to run again in the future. i2 Analyze provides a way to save these artifacts, and also to share them with other users.

- All users of i2 Analyze deployments that include the Information Store can save artifacts for reuse.
- If they have the appropriate permission, users can share saved artifacts with other users of the same deployment. Artifacts can be shared with individuals or with groups of users.
- All users can see artifacts that have been shared with them, with no additional permissions necessary.

When a user first saves an artifact, they become its sole *owner*. An artifact owner can decide to share it with other users and groups. When they do so, they can say what those users and groups can do with the artifact:

- A *viewer* of an artifact can see it, use it, and make a copy of it - but they can't save changes to the original artifact or affect how it's shared.
- An *editor* of an artifact can do everything a viewer can. They can also change the artifact itself, and assign viewer and editor permissions to other users and groups.
- An owner of an artifact can do everything an editor can. They can also appoint other users and groups as owners. Only users with owner permissions can delete an artifact.

Note: The permission that allows a user to share an artifact overrides the "editor" and "owner" settings. In other words, a user who can't share artifacts is *always* treated as a "viewer".

Configuration settings

Saving and sharing artifacts is enabled and controlled through several different configuration settings:

- For an i2 Analyze user to share artifacts, they must be a member of a system group that has the appropriate command access control permission.
- For members of a group to have artifacts shared with them, that group must be made available for sharing in the admin console. You can also use the admin console to create user groups for this purpose.
- Users manage shared artifacts through the i2 Analyze client applications: i2 Notebook on the web, and Analyst's Notebook on the desktop.

At your discretion, you can also configure system user groups with a special permission that allows members to modify the sharing settings of *all* saved artifacts, regardless of ownership. For more information, see Administrator permissions.

Logging and auditing

i2 Analyze provides mechanisms for logging two types of information that the system generates during normal execution. You can control what information is sent to the system logs, and audit the commands that users invoke.

System logging

The components that make up the i2 Analyze server all contain instrumentation that sends information about the health of the system to log files or the console. You can control the locations of the log files, and the volume of information that the system sends, by editing the `log4j2.xml` files in the deployment toolkit.

The information that i2 Analyze can log through this mechanism includes detail about warnings and errors that users see in their client software, and incremental status reports about long-running processes such as ingestion.

The ZooKeeper component of i2 Analyze uses Logback for logging. The Logback configuration is in the `i2analyze\deploy\zookeeper\conf` directory.

For more information about system logging, see the deployment and configuration guides for i2 Analyze, or the Apache Log4j website.

User activity logging

When a user runs an authenticated command against any of its services, i2 Analyze can record information about the user who ran the command, and full details of the command that they ran. For example, you might use this functionality to audit the frequency with which different users make requests for the data that i2 Analyze manages, or to track searches with particular patterns. i2 Analyze handles user activity logging for the i2 Connect gateway separately from the Information Store and the Chart Store.

Note: Depending on the volume of data, enabling user activity logging might affect the performance of i2 Analyze.

Information Store and Chart Store

i2 Analyze supports user activity logging for all of the main analysis operations against the Information Store and (where relevant) the Chart Store. For example, you can configure separate logging (or no

logging at all) for search, expand, and find path operations. You can also arrange for logging to occur when records and charts are created or modified.

To audit user activity, including activity due to Analyst's Notebook and the Investigate Add-On, you write a class that implements the `IAuditLogger` interface and specify it in the `ApolloServerSettingsMandatory.properties` file in the deployment toolkit.

At startup, i2 Analyze calls `IAuditLogger` to discover what activities to log information about. Later, it calls again with information such as the time of the activity, the name and security clearance of the user, and the parameters that they supplied.

For more information and an example of how to implement `IAuditLogger`, see i2 Analyze Developer Essentials.

i2 Connect gateway

To log operations against external sources through the i2 Connect gateway, i2 Analyze uses the same `IAuditLogger` interface that it uses for the Information Store and chart store. However, all such operations are logged through a single method on the `IAuditLogger` interface.

Deploying i2 Analyze

All deployments of i2 Analyze are different in terms of the functionality they support and the components they employ. They are also different in terms of how you intend them to be used. The process by which you deploy i2 Analyze changes significantly, depending on whether your target is an example or a production environment.

Deployment information

- Deployment types

Deployment tasks

- Creating an example deployment
- Creating a production deployment

Troubleshooting and support

- i2 Analyze support page
- i2 Support

Deployment types

Before you start a deployment of i2 Analyze, choose the type of deployment that you want to create. Use the following information to ensure that you choose the correct type of deployment for your requirements.

Example deployment

You can use an example deployment to learn about i2 Analyze, demonstrate the features of the system, and ensure that any software prerequisites are installed correctly on a single server.

When you create an example deployment, the deployment toolkit populates all of the mandatory configuration settings with default values and deploys the system. The deployment uses an example i2 Analyze schema, security schema, and data. Some configuration settings that are not mandatory for deployment are also populated to demonstrate extra features of the system.

For more information about example deployments, see Creating an example deployment.

Production deployment

A production deployment is available to analysts to complete mission critical analysis on real-world data. When you decide to create a production deployment of i2 Analyze, you must start from a clean installation of i2 Analyze.

The process for creating a production deployment involves a number of different deployment and configuration activities. As part of the process, you must develop an i2 Analyze schema and security schema for your data.

For more information about production deployments, see Creating a production deployment.

Creating an example deployment

To understand what i2 Analyze is, and to demonstrate the features of the system, you can create an example deployment.

An example deployment uses default values that are provided by the deployment toolkit, and contains a configuration that demonstrates the features provided by i2 Analyze. You can also use an example deployment to verify that any prerequisites are installed correctly.

Creating an example with the Chart Store

An installation of i2 Analyze includes example settings for deploying the server with the Chart Store. This topic describes how to use those settings in an example deployment.

Before you begin

Install i2 Analyze and any software prerequisites. For more information, see Installing i2 Analyze. To deploy the Chart Store, you need PostgreSQL, IBM Db2, or Microsoft SQL Server.

If you are using PostgreSQL, which is the default option, download the latest JDBC driver from jdbc.postgresql.org. The name of the driver file has the form `postgresql-x.y.z.jar`.

If you are using SQL Server, download the latest version of the Microsoft JDBC Driver for SQL Server that's compatible with your system from learn.microsoft.com/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server. Extract the contents of the download, and locate the file whose name matches the pattern `mssql-jdbc-x.y.z.jre11.jar`.

Important: For a deployment of Analyst's Notebook to a small workgroup, an example deployment of i2 Analyze with the Chart Store and custom security settings might offer sufficient performance for use in a production environment. For a larger workgroup, or for a group with a large number of charts, a production deployment is appropriate.

About this task

The following procedure describes how to create an example deployment of i2 Analyze with the Chart Store. The i2 Analyze toolkit contains an example configuration for the deployment. The `deployExample` task generates the default values for the mandatory settings and deploys the platform.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in.

In the example deployment, i2 Analyze runs with an example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is `Jenny`
- The password is `Jenny`

The example deployment uses the `chart-storage-schema.xml` schema file, with the associated `chart-storage-schema-charting-schemes.xml` file as the charting scheme.

Procedure

First, create and populate the configuration directory:

1. Navigate to the `\toolkit\examples\configurations\chart-storage` directory.

   This directory contains the preconfigured files that you need to deploy a system that uses the Chart Store to store Analyst's Notebook charts.
2. Copy the `configuration` directory to the root of the toolkit. For example, `C:\i2\i2analyze\toolkit\configuration`.
3. If you are using PostgreSQL as your database management system, copy the PostgreSQL JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
4. If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example:
   a. Copy the example `topology.xml` file for SQL Server from `toolkit\configuration\examples\topology\sqlserver` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
   b. Copy the SQL Server JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
5. If you are using IBM Db2 as your database management system, copy the example `topology.xml` file for IBM Db2 from `toolkit\configuration\examples\topology\db2` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.

Then, regardless of your database management system, you must complete the following steps after you create the configuration directory.

1. Specify the credentials to use for the deployment.
   a. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter the user name and password to use with the database.
   c. Enter a user name and password to use for Solr.
   d. Enter a password to use to encrypt LTPA tokens.
   e. Save and close the `credentials.properties` file.
2. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the `topology.xml` file.

   By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see Db2 server TCP/IP port numbers.
   a. Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   b. In the `<database>` element, set the value of the `port-number` attribute to `25000`.

c. Save and close the `topology.xml` file.

3. Run the setup script to create the example deployment.
   a. Open a command prompt and navigate to the `toolkit\scripts` directory.
   b. To deploy the example, run the following command:

      ```
      setup -t deployExample
      ```

   c. To start the Liberty server, run the following command:

      ```
      setup -t start
      ```

What to do next

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

```
Web application available (default_host): http://<host_name>:9082/opal
```

Install Analyst's Notebook and connect to your deployment. For more information, see Connecting clients.

Creating an example with the Information Store

An installation of i2 Analyze includes example settings for deploying the server with the Information Store. This topic describes how to use those settings in an example deployment.

Before you begin

Install i2 Analyze and any software prerequisites. For more information, see Installing i2 Analyze. To deploy the preconfigured examples for the Information Store, you need PostgreSQL, IBM Db2, or Microsoft SQL Server.

If you are using PostgreSQL, which is the default option, download the latest JDBC driver from jdbc.postgresql.org. The name of the driver file has the form `postgresql-x.y.z.jar`.

If you are using SQL Server, download the latest version of the Microsoft JDBC Driver for SQL Server that's compatible with your system from learn.microsoft.com/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server. Extract the contents of the download, and locate the file whose name matches the pattern `mssql-jdbc-x.y.z.jre11.jar`.

About this task

The following procedure describes how to create an example deployment of i2 Analyze with the Information Store, which also fulfills the functions of the Chart Store. The i2 Analyze toolkit contains an example configuration for the deployment. The `deployExample` task generates the default values for the mandatory settings and deploys the platform.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in.

In the example deployment, i2 Analyze runs with an example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is `Jenny`
- The password is `Jenny`

The example deployment uses the `law-enforcement-schema.xml` schema file as the i2 Analyze schema, with the associated `law-enforcement-schema-charting-schemes.xml` as the charting scheme.

Procedure

First, create and populate the configuration directory:

1. Navigate to the `\toolkit\examples\configurations\information-store-opal` directory.

   This directory contains the preconfigured files that you need to deploy a system that uses the Information Store to store data and Analyst's Notebook charts, and to support the i2 Notebook web client.
2. Copy the `configuration` directory to the root of the toolkit. For example, `C:\i2\i2analyze\toolkit\configuration`.
3. If you are using PostgreSQL as your database management system, copy the PostgreSQL JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
4. If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example:
   a. Copy the example `topology.xml` file for SQL Server from `toolkit\configuration\examples\topology\sqlserver` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
   b. Copy the SQL Server JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
5. If you are using IBM Db2 as your database management system, copy the example `topology.xml` file for IBM Db2 from `toolkit\configuration\examples\topology\db2` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.

Then, regardless of your database management system, you must complete the following steps after you create the configuration directory.

1. Specify the credentials to use for the deployment.
   a. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter the user name and password to use with the database.
   c. Enter a user name and password to use for Solr.
   d. Enter a password to use to encrypt LTPA tokens.
   e. Save and close the `credentials.properties` file.
2. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the `topology.xml` file.

   By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see Db2 server TCP/IP port numbers.
   a. Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   b. In the `<database>` element, set the value of the `port-number` attribute to `25000`.
   c. Save and close the `topology.xml` file.

3. Run the setup script to create the example deployment.
   a. Open a command prompt and navigate to the `toolkit\scripts` directory.
   b. To deploy the example, run the following command:
      ```
      setup -t deployExample
      ```
   c. To start the Liberty server, run the following command:
      ```
      setup -t start
      ```
4. **Optional:** To populate your Information Store with the provided example data for the `law-enforcement-schema.xml` schema, run the following command:
   ```
   setup -t ingestExampleData
   ```

What to do next

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

```
Web application available (default_host): http://<host_name>:9082/opal
```

Install Analyst's Notebook or open a web browser and connect to your deployment. For more information, see Connecting clients.

Creating an example with the i2 Connect gateway

An installation of i2 Analyze includes example settings for deploying the server with support for the i2 Connect gateway only. With these settings, the i2 Connect gateway enables Analyst's Notebook users to search for and retrieve data from external data sources, and then to analyze the results on charts.

Before you begin

Install i2 Analyze and any software prerequisites. For more information, see Installing i2 Analyze.

Before you create the example deployment, you must download and install Node.js to host the example connector. Download Node.js for your operating system from https://nodejs.org/en/download. You can install Node.js with the default settings.

About this task

To use any deployment of i2 Analyze with the i2 Connect gateway, you must obtain or create a connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and i2 publishes a package that contains an example connector to www.npmjs.com.

The example deployment demonstrates a working i2 Analyze system that can query and retrieve data from an external data source. You can log in with an example user. In the example deployment, i2 Analyze runs with the example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is `Jenny`
- The password is `Jenny`

Note: This example does not support the i2 Notebook web client. To give users the option of using the web client, you must Create an example with the Chart Store.

Procedure

1. Create the configuration directory:
   a. Navigate to the `\toolkit\examples\configurations\daod-opal` directory.

      This directory contains the preconfigured files that you need to deploy a system that uses the i2 Connect gateway to connect to an external data source.
   b. Copy the `configuration` directory to the root of the toolkit. For example, `C:\i2\i2analyze\toolkit\configuration`.
2. Specify the credentials to use for the deployment.
   a. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter a user name and password to use for Solr.
   c. Enter a password to use to encrypt LTPA tokens.
   d. Save and close the `credentials.properties` file.
3. Run the setup script to create the example deployment.
   a. Open a command prompt and navigate to the `toolkit\scripts` directory.
   b. To deploy the example, run the following command:

      ```
      setup -t deployExample
      ```
4. Install the dependencies and start the server that hosts the example connector.

   Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.
   a. Open another command prompt, separate from the one where you ran the `deployExample` command.
   b. To download and run the package that contains the example connector, execute the following command:

      ```
      npx @i2analyze/example-connector
      ```

      Note: You must be connected to the internet to download the package.
5. Start i2 Analyze.
   a. Return to the command prompt where you ran the `deployExample` command.
   b. To start i2 Analyze, run the following command:

      ```
      setup -t start
      ```

What to do next

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

```
Web application available (default_host): http://<host_name>:9082/opaldaod
```

Install Analyst's Notebook and connect to your deployment. For more information, see Connecting clients.

Production deployments of i2 Analyze use client-authenticated SSL communication between i2 Analyze and any connectors. The example deployment does not use it, and so Analyst's Notebook displays a warning to that effect when you open the external searches window. For more information about

configuring client-authenticated SSL, see Client-authenticated Secure Sockets Layer with the i2 Connect gateway.

You can create your own connectors to use with the deployment of i2 Analyze. For more information, see i2 Analyze and the i2 Connect gateway.

Creating an example with the Chart Store and the i2 Connect gateway

An installation of i2 Analyze includes example settings for deploying the server with the Chart Store and support for the i2 Connect gateway. With these settings, Analyst's Notebook users can upload charts to the Chart Store, while both they and i2 Notebook web client users can search for and retrieve data from an example external data source through the i2 Connect gateway.

Before you begin

Install i2 Analyze and any software prerequisites. For more information, see Installing i2 Analyze. To deploy the Chart Store, you need PostgreSQL, IBM Db2, or Microsoft SQL Server.

If you are using PostgreSQL, which is the default option, download the latest JDBC driver from jdbc.postgresql.org. The name of the driver file has the form `postgresql-x.y.z.jar`.

If you are using SQL Server, download the latest version of the Microsoft JDBC Driver for SQL Server that's compatible with your system from learn.microsoft.com/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server. Extract the contents of the download, and locate the file whose name matches the pattern `mssql-jdbc-x.y.z.jre11.jar`.

Before you create the example deployment, you must download and install Node.js to host the example connector. Download Node.js for your operating system from https://nodejs.org/en/download. You can install Node.js with the default settings.

About this task

The following procedure describes how to create an example deployment of i2 Analyze with the Chart Store and the i2 Connect gateway. To use any deployment of i2 Analyze with the i2 Connect gateway, you must obtain or create a connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and i2 publishes a package that contains an example connector to www.npmjs.com.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in. In the example deployment, i2 Analyze runs with an example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is `Jenny`
- The password is `Jenny`

The example deployment uses the `chart-storage-schema.xml` schema file, with the associated `chart-storage-schema-charting-schemes.xml` file as the charting scheme.

Procedure

First, create and populate the configuration directory:

1. Navigate to the `\toolkit\examples\configurations\chart-storage-daod` directory.

   This directory contains the preconfigured files that you need to deploy a system that uses the i2 Connect gateway to connect to an external data source, and the Chart Store to store Analyst's Notebook charts and support the i2 Notebook web client.

2. Copy the `configuration` directory to the root of the toolkit. For example, `C:\i2\i2analyze\toolkit\configuration`.
3. If you are using PostgreSQL as your database management system, copy the PostgreSQL JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
4. If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example:
   a. Copy the example `topology.xml` file for SQL Server from `toolkit\configuration\examples\topology\sqlserver` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
   b. Copy the SQL Server JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
5. If you are using IBM Db2 as your database management system, copy the example `topology.xml` file for IBM Db2 from `toolkit\configuration\examples\topology\db2` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.

Then, regardless of your database management system, you must complete the following steps after you create the configuration directory.

1. Specify the credentials to use for the deployment.
   a. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter the user name and password to use with the database.
   c. Enter a user name and password to use for Solr.
   d. Enter a password to use to encrypt LTPA tokens.
   e. Save and close the `credentials.properties` file.
2. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the `topology.xml` file.

   By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see Db2 server TCP/IP port numbers.
   a. Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   b. In the `<database>` element, set the value of the `port-number` attribute to `25000`.
   c. Save and close the `topology.xml` file.
3. Run the setup script to create the example deployment.
   a. Open a command prompt and navigate to the `toolkit\scripts` directory.
   b. To deploy the example, run the following command:

      ```
      setup -t deployExample
      ```
4. Install the dependencies and start the server that hosts the example connector.

   Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.
   a. Open another command prompt, separate from the one where you ran the `deployExample` command.

b. To download and run the package that contains the example connector, execute the following command:

```
npx @i2analyze/example-connector
```

Note: You must be connected to the internet to download the package.

5. Start i2 Analyze.

a. Return to the command prompt where you ran the `deployExample` command.

b. To start i2 Analyze, run the following command:

```
setup -t start
```

What to do next

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

```
Web application available (default_host): http://<host_name>:9082/opal
```

Install Analyst's Notebook or open a web browser and connect to your deployment. For more information, see Connecting clients.

Production deployments of i2 Analyze use client-authenticated SSL communication between i2 Analyze and any connectors. The example deployment does not use it, and so Analyst's Notebook displays a warning to that effect when you open the external searches window. For more information about configuring client-authenticated SSL, see Client-authenticated Secure Sockets Layer with the i2 Connect gateway.

You can create your own connectors to use with the deployment of i2 Analyze. For more information, see i2 Analyze and the i2 Connect gateway.

Creating an example with the Information Store and the i2 Connect gateway

An installation of i2 Analyze includes example settings for deploying the server with the Information Store and support for the i2 Connect gateway. Users can access data in the Information Store in the usual manner, and the i2 Connect gateway enables analysts to search for and retrieve data from an example external data source.

Before you begin

Install i2 Analyze and any software prerequisites. For more information, see Installing i2 Analyze. To deploy the preconfigured examples for the Information Store, you need PostgreSQL, IBM Db2, or Microsoft SQL Server.

If you are using PostgreSQL, which is the default option, download the latest JDBC driver from jdbc.postgresql.org. The name of the driver file has the form `postgresql-x.y.z.jar`.

If you are using SQL Server, download the latest version of the Microsoft JDBC Driver for SQL Server that's compatible with your system from learn.microsoft.com/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server. Extract the contents of the download, and locate the file whose name matches the pattern `mssql-jdbc-x.y.z.jre11.jar`.

Before you create the example deployment, you must download and install Node.js to host the example connector. Download Node.js for your operating system from https://nodejs.org/en/download. You can install Node.js with the default settings.

About this task

The following procedure describes how to create an example deployment of i2 Analyze with the Information Store, which also fulfills the functions of the Chart Store, and the i2 Connect gateway. To use any deployment of i2 Analyze with the i2 Connect gateway, you must obtain or create a connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and i2 publishes a package that contains an example connector to www.npmjs.com.

The example deployment demonstrates a working i2 Analyze system with an example user so that you can log in. You can ingest example data into the Information Store and then perform searches and analysis, and use the example connector to query and retrieve data from an external data source.

In the example deployment, i2 Analyze runs with an example security schema and matching Liberty security groups and users. The example user has the following credentials:

- The user name is `Jenny`
- The password is `Jenny`

The example deployment uses the `law-enforcement-schema.xml` schema file as the i2 Analyze schema, with the associated `law-enforcement-schema-charting-schemes.xml` as the charting scheme.

Procedure

First, create and populate the configuration directory:

1. Navigate to the `\toolkit\examples\configurations\information-store-daod-opal` directory.

   This directory contains the preconfigured files that you need to deploy a system that uses the i2 Connect gateway to connect to an external data source, and the Information Store to store data and Analyst's Notebook charts, and to support the i2 Notebook web client.
2. Copy the `configuration` directory to the root of the toolkit. For example, `C:\i2\i2analyze\toolkit\configuration`.
3. If you are using PostgreSQL as your database management system, copy the PostgreSQL JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
4. If you are using SQL Server as your database management system, you must complete extra configuration actions to deploy the example:
   a. Copy the example `topology.xml` file for SQL Server from `toolkit\configuration\examples\topology\sqlserver` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
   b. Copy the SQL Server JDBC driver file that you downloaded to the `toolkit\configuration\environment\common\jdbc-drivers` directory.
5. If you are using IBM Db2 as your database management system, copy the example `topology.xml` file for IBM Db2 from `toolkit\configuration\examples\topology\db2` to the `toolkit\configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.

Then, regardless of your database management system, you must complete the following steps after you create the configuration directory.

1. Specify the credentials to use for the deployment.

a. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
b. Enter the user name and password to use with the database.
c. Enter a user name and password to use for Solr.
d. Enter a password to use to encrypt LTPA tokens.
e. Save and close the `credentials.properties` file.

2. If you are using IBM Db2 11.5.6 Fix Pack 0 or later, you might need to update the port number that is used in the `topology.xml` file.

   By default, i2 Analyze is deployed to connect to Db2 using port 50000. In version 11.5.6 Fix Pack 0 and later, the default port that Db2 uses is changed to 25000. For more information about the ports that Db2 uses, see Db2 server TCP/IP port numbers.

   a. Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   b. In the `<database>` element, set the value of the `port-number` attribute to `25000`.
   c. Save and close the `topology.xml` file.

3. Run the setup script to create the example deployment.

   a. Open a command prompt and navigate to the `toolkit\scripts` directory.
   b. To deploy the example, run the following command:

      ```
      setup -t deployExample
      ```

4. Install the dependencies and start the server that hosts the example connector.

   Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

   a. Open another command prompt, separate from the one where you ran the `deployExample` command.
   b. To download and run the package that contains the example connector, execute the following command:

      ```
      npx @i2analyze/example-connector
      ```

      Note: You must be connected to the internet to download the package.

5. Start i2 Analyze.

   a. Return to the command prompt where you ran the `deployExample` command.
   b. To start i2 Analyze, run the following command:

      ```
      setup -t start
      ```

6. **Optional:** To populate your Information Store with the provided example data for the `law-enforcement-schema.xml` schema, run the following command:

   ```
   setup -t ingestExampleData
   ```

What to do next

When you start i2 Analyze, the URI that you can use to connect to the deployment is displayed in the console. For example:

```
Web application available (default_host): http://<host_name>:9082/opal
```

Install Analyst's Notebook or open a web browser and connect to your deployment. For more information, see Connecting clients.

Production deployments of i2 Analyze use client-authenticated SSL communication between i2 Analyze and any connectors. The example deployment does not use it, and so Analyst's Notebook displays a warning to that effect when you open the external searches window. For more information about configuring client-authenticated SSL, see Client-authenticated Secure Sockets Layer with the i2 Connect gateway.

You can create your own connectors to use with the deployment of i2 Analyze. For more information, see i2 Analyze and the i2 Connect gateway.

Creating a production deployment

The process of creating a production deployment is separated into a number of different activities, which you complete in an iterative process. The suggested process involves the creation and retention of several environments, each one focused on different aspects of a production deployment of i2 Analyze.

Planning for production

Before you can start to install and deploy i2 Analyze, you must first understand how i2 Analyze fits into your organization:

- Understand what i2 Analyze is.
- Understand the requirements of the deployment, and align these to a deployment pattern.
- Understand the data and security models of the environment that i2 Analyze is deployed in.

For more information, see the Understanding section.

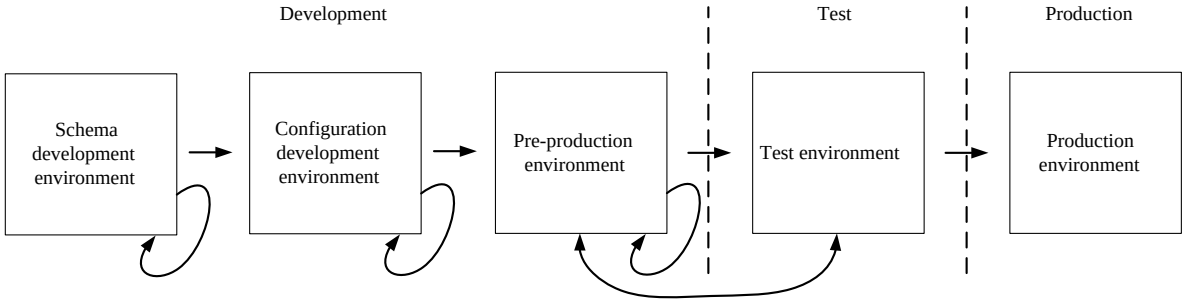
Deploying

After you identify the requirements of the deployment, you can start to create the production deployment. The process of deploying i2 Analyze is completed in three phases that are explained in Deployment phases and environments.

Note: If your planned deployment of i2 Analyze is to a small workgroup and includes only the Chart Store, an example deployment with the Chart Store and custom security settings might offer sufficient performance. For more information about example deployments, see Creating an example with the Chart Store.

Deployment phases and environments

The process of creating a production deployment involves distinct phases that focus on different aspects of an i2 Analyze deployment. You should complete the activities in each phase in distinct environments, which you should retain to refer to later in the process.



There are three phases in the process to deploy i2 Analyze into production.

Development

The *development phase* is where you configure i2 Analyze to meet the requirements of the final deployment. In this phase, you develop the configuration in an iterative process that involves a number of configuration changes and deployments of the system. During this phase, the lifetime of a deployment is short.

Test

The *test phase* is where you deploy i2 Analyze for testing. In the test phase, you deploy i2 Analyze with the configuration from the development phase and perform comprehensive testing of the system.

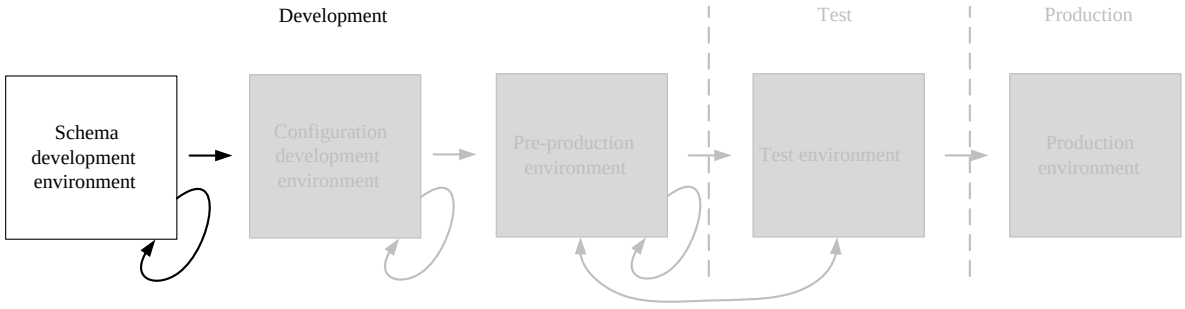
Production

The *production phase* is where you deploy i2 Analyze into production. i2 Analyze is deployed with the configuration that you tested in the test phase. In production, the deployment is fully operational and used by analysts to complete mission critical work.

To start creating your production deployment, complete the instructions in Schema development environment.

Schema development environment

The first task in the development phase is often to develop the schema and the security schema for the deployment. You can use a *schema development environment*, which features a single-server deployment of i2 Analyze with the i2 Connect gateway to develop the schemas.



In an environment that includes a database, significant changes to the schema or the security schema can be time-consuming. Destructive changes to either schema require you to rebuild the database. The purpose of the schema development environment is to enable rapid iteration. When you apply the schemas that you create here to the Information Store in the configuration development environment, they are less likely to need significant changes.

Creating the schema development environment

In the schema development environment, you first populate the values of some of the mandatory settings for deploying i2 Analyze. After you deploy i2 Analyze for the first time, you can then develop the schemas to meet your requirements.

Before you begin

The easiest approach is to use the containerized configuration development environment as your *schema development environment* and *configuration development environment*. The containerized environment provides functionality that makes it easier to develop your i2 Analyze configuration, including:

- No requirement to install a database management system locally
- Improved experience for editing configuration files
- A simplified deployment process

After you develop your configuration in the containerized development environment, you can export your configuration to use with your on-premise deployment toolkit in the *pre-production*, *test*, and *production* environments.

If you do not want to use the containerized environment, continue to complete the following instructions.

About this task

Depending on the terms of your license and the needs of your organization, a deployment of i2 Analyze might include the Chart Store alone, or the Information Store alone, or the i2 Connect gateway. The

i2 Connect gateway can itself appear alone, or in combination with the Chart Store or the Information Store.

- If your target deployment includes only the Chart Store, then *you do not need to create a schema development environment*. Instead, you can go directly to the configuration development environment.
- If your target deployment does not include the Information Store but does include the i2 Connect gateway, then the schema development environment is a convenient place to develop a gateway schema. If you do not intend to use a gateway schema, the environment still provides support for prototyping a security schema.
- If your target deployment includes the Information Store, then you must complete the steps in this and subsequent topics to develop the schema that will become your Information Store schema.

The schema development environment features an instance of i2 Analyze with the i2 Connect gateway; there is no Information Store or Chart Store. By using a deployment without a database, you can quickly prototype changes to the schema and the security schema, and then visualize the changes in Analyst's Notebook.

Procedure

1. Install any prerequisite software to prepare your server for the schema development environment.
   - For the schema development environment, prepare your server for the i2 Connect deployment topology.
   - Install Analyst's Notebook to connect to your deployment and i2 Analyze Schema Designer to edit your schema files. For more information, see Installing i2 Analyst's Notebook and Working with i2 Analyze.
2. In the deployment toolkit that you installed, copy the `toolkit\examples\configurations\daod-opal\configuration` directory to the `toolkit` directory.
3. Enter a user name and password to use with the Solr index and a password to encrypt the LTPA tokens in the `configuration\environment\credentials.properties` file. For more information about credentials in i2 Analyze, see Specifying the deployment credentials.
4. In a command prompt, navigate to the `toolkit\scripts` directory, and run the following command to populate some mandatory settings with default values:

   ```
   setup -t generateDefaults
   ```

   The `environment.properties` and `topology.xml` are modified by this toolkit task. For more information about the default values that are provided, see Configuration files reference.
5. In i2 Analyze Schema Designer, either create a new schema or open one of the examples to modify.

   For information about creating or modifying schema files, see Creating schemas and Charting schemes.

   Example schema files are located in subdirectories of the `toolkit\examples\` directory. For more information, see Example schemas.

   Note: If your planned deployment does not include the Information Store and you do not intend to use a gateway schema, your choice of example here is not important. You just need to set up a valid development environment.

   a. Save the initial version of your schema in the `configuration\fragments\common\WEB-INF\classes` directory. A charting scheme file is saved in the same location when you save the schema.

b. Keep the schema file open in Schema Designer so that you can make more modifications after you deploy i2 Analyze.

6. Copy the example security schema to the `configuration\fragments\common\WEB-INF\classes` directory. The example security schema file is located in the `toolkit\configuration\examples\security-schema` directory. For more information, see Example schemas.
7. In the `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties` file, set the values of the following settings to the file names of your schema, charting scheme, and security schema:
   - `Gateway.External.SchemaResource`
   - `Gateway.External.ChartingSchemesResource`
   - `DynamicSecuritySchemaResource`

   For example, `Gateway.External.SchemaResource=custom-schema.xml`
8. Deploy i2 Analyze:

   ```
   setup -t deploy
   ```
9. Create an example user that you can use to log in:

   ```
   setup -t ensureExampleUserRegistry
   ```

   The user has the user name 'Jenny' and the password 'Jenny'.
10. Start i2 Analyze:

    ```
    setup -t startLiberty
    ```

    When you start i2 Analyze, the URI that you can use to connect to it from Analyst's Notebook is displayed in the console. For example: `Web application available (default_host): http://host_name:9082/opaldaod`
11. Connect to your deployment by using Analyst's Notebook. Log in with the example 'Jenny' user.
12. In Analyst's Notebook, create items on the chart to visualize the i2 Analyze schema by using the **Gateway** palette.

What to do next

After you deploy i2 Analyze with the initial schema files, you can develop them for your own data requirements:

- Updating the schema and the charting scheme
- Updating the security schema

After you develop your schema files, you move to the configuration development environment. In the configuration development environment, you deploy i2 Analyze with the schemas that you developed in your schema development environment.

You can choose whether to retain your schema development environment. If you need to make small changes to your schemas later, you can do this in your configuration development environment. If you need to make more substantial changes, you might create another schema development environment.

When you have finalized your schemas, you can move to the next environment. For more information, see Configuration development environment.

Updating the schema and the charting scheme

In the schema development environment, you can quickly deploy your changes to the schema and the charting scheme. Use this environment to develop the schema and charting scheme for your production deployment.

About this task

Develop a schema and an associated charting scheme to meet your data requirements. Then update your schema development environment with the modified schema and charting scheme.

Procedure

1. In i2 Analyze Schema Designer, modify the schema that is deployed in your schema development environment. For information about creating or modifying your schema files, see Creating schemas and Charting schemes.

After you modify your schema, update the deployment with your changes.

1. Update and redeploy the system:

   ```
   setup -t updateConnectorsConfiguration
   ```
2. Start i2 Analyze:

   ```
   setup -t restartLiberty
   ```
3. Test the changes to the schema and charting scheme by connecting to your deployment in Analyst's Notebook and modeling representative data on the chart by using the **Gateway** palette.

What to do next

Repeat this process until the schema and the charting scheme meet your requirements. When your schema development is complete, store your schema and charting scheme files in a version control system.

This is not the final chance to modify the schema, but it should now contain all the entity and link types that you require, and most of the property types.

When you are satisfied with the schema, develop the security schema for the deployment. For more information, see Updating the security schema.

Updating the security schema

In the schema development environment, you can quickly make and deploy changes to the i2 Analyze security schema. Use this environment to develop the security schema for your production deployment.

About this task

Develop a security schema to meet your security requirements. Then update your schema development environment with the modified security schema.

Procedure

1. In an XML editor, either create a new security schema or open the one that is deployed in the schema development environment. For information about creating or modifying a security schema file for i2 Analyze, see The i2 Analyze Security schema.

a. Save your security schema in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.

b. Ensure that your security schema file is specified in `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties`.

After you modify your security schema, update the deployment with your changes.

1. Update and redeploy the system:

```
setup -t updateSecuritySchema
setup -t deployLiberty
```

2. Start i2 Analyze:

```
setup -t restartLiberty
```

3. If you changed the names of the user groups in the security schema, update the basic user registry to match the new names. For more information, see Configuring the Liberty user registry.

4. Test the changes to the security schema by connecting to the deployment in Analyst's Notebook as different users and changing the security permissions on records that you create.

What to do next

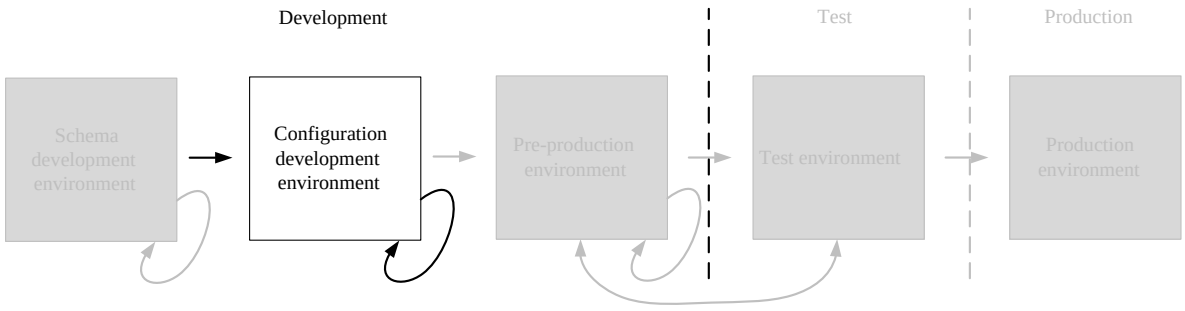
Repeat this process until your security schema meets your requirements. When your security schema development is complete, store your security schema and Liberty user registry files in a version control system.

This is not the final time that you can modify the security schema, but you should aim to have most of the security dimensions and dimension values defined.

After you finish developing your schema files, you can move to the next environment. For more information, see Configuration development environment.

Configuration development environment

In the configuration development environment, you configure i2 Analyze to meet your requirements. The aspects of i2 Analyze that you might configure include how your data is added to the system and how analysts interact with that data.



After you create your schemas, you can use a *configuration development environment* to configure i2 Analyze in a single server environment. In the configuration development environment, you deploy i2 Analyze with the same data store as your intended production system.

In this deployment of i2 Analyze, you use any schema files that you previously created. If your intended production system includes the Information Store and you have not created your schema files, complete the instructions in Schema development environment.

Creating the configuration development environment

When you create the configuration development deployment, you deploy i2 Analyze on a single server in the same deployment pattern as your intended production system.

Before you begin

The easiest approach is to use the containerized configuration development environment as your *schema development environment* and *configuration development environment*. The containerized environment provides functionality that makes it easier to develop your i2 Analyze configuration, including:

- No requirement to install a database management system locally
- Improved experience for editing configuration files
- A simplified deployment process

After you develop your configuration in the containerized development environment, you can export your configuration to use with your on-premise deployment toolkit in the *pre-production*, *test*, and *production* environments.

If you intend your production deployment to contain only the i2 Connect gateway, and to be accessed only through Analyst's Notebook, you can use your schema development environment as your configuration development environment. In this scenario, you can move to develop the i2 Analyze configuration.

If your intended production deployment contains the Information Store, ensure that you have access to the following files from the schema development environment so that you can copy them to the configuration development environment:

- The Information Store schema
- The associated charting scheme
- The security schema
- The user registry, `user-registry.xml`

If you did not create a schema development environment because you intend your production environment to contain the Chart Store, you must retrieve equivalent files from the deployment toolkit:

- The Chart Store schema, `chart-storage-schema.xml`
- The associated charting scheme, `chart-storage-schema-charting-schemes.xml`
- The example security schema, `example-dynamic-security-schema.xml`

For deployments that include the Chart Store, the following procedure explains how to create an example user registry when it becomes necessary to do so.

Procedure

1. Install any prerequisite software to prepare your server for the configuration development environment.
   - For the configuration development environment, prepare your server for the Single server deployment topology with the same components and database management system as in your intended production system.
2. Create the configuration directory in the i2 Analyze deployment toolkit.
   a. Navigate to the `toolkit\examples\configurations` directory where you installed i2 Analyze.
   b. Copy the `configuration` directory from your chosen base configuration to the root of the toolkit. For example, copy the `toolkit\examples\configurations\information-store-opal\configuration` directory to the `toolkit` directory. For more information about the different base configurations, see The base configurations.
3. If you are using SQL Server as your database management system, copy the example `topology.xml` file from `configuration\examples\topology\sqlserver` to the `configuration\environment` directory. Overwrite the existing `topology.xml` file in the destination directory.
4. If you are using IBM HTTP Server, in the `topology.xml` file set the value of the `http-server-host` attribute to `true`.

To allow the deployment toolkit to create and modify components of i2 Analyze, you provide user names and passwords for each of the components in your configuration.

1. Specify the user names and passwords to use for the deployment in the `toolkit\configuration\environment\credentials.properties` file. For more information about credentials in i2 Analyze, see Specifying the deployment credentials.
2. Copy the JDBC driver to use with the deployment to the `configuration\environment\common\jdbc-drivers` directory. For more information, see Specifying the JDBC driver. You do not need to provide a JDBC driver for deployment that contains only the i2 Connect gateway.
3. Use the deployment toolkit to populate some of the mandatory settings with default values:

   ```
   setup -t generateDefaults
   ```

   The `environment.properties` and `topology.xml` are modified by this toolkit task. For more information about the default values that are provided, see Configuration files reference.
4. Specify the i2 Analyze schema, charting scheme, and security schema that you previously prepared.
   a. Copy your i2 Analyze schema, charting scheme, and security schema files to the `configuration\fragments\common\WEB-INF\classes` directory.
   b. In the `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties` file, set the file names of the schema, charting scheme, and security schema.
      - If your deployment includes the Information Store or the Chart Store, set `SchemaResource` and `ChartingSchemesResource`.
      - If your deployment includes the i2 Connect gateway and you have developed a gateway schema, set `Gateway.External.SchemaResource` and `Gateway.External.ChartingSchemesResource`
      - In both cases, set `DynamicSecuritySchemaResource`

5. Deploy i2 Analyze:

   ```
   setup -t deploy
   ```

6. If you created a schema development environment, copy the user registry file from that environment to the `deploy\wlp\usr\shared\config` directory in the new environment.

   If you did not create a schema development environment, execute the following command to create an example user that you can use to log in:

   ```
   setup -t ensureExampleUserRegistry
   ```

   The user has the user name 'Jenny' and the password 'Jenny'.

7. Start i2 Analyze:

   ```
   setup -t start
   ```

   The URI that users must specify is displayed in the console. For example:

   ```
   Web application available (default_host): http://host_name:9082/opal/
   ```

What to do next

After you deploy i2 Analyze, you can begin Developing the i2 Analyze configuration.

Developing the i2 Analyze configuration

At this stage of the production deployment process, you can use the configuration development environment to define how the i2 Analyze application behaves specifically for your organization.

About this task

Developing the configuration is separated into two parts: enabling i2 Analyze to work with your data, and defining how analysts interact with your data.

The i2 Analyze deployment in the configuration development environment is not your final deployment. When you are configuring the deployment, use small amounts of data with a few users to ensure that the processes are functional.

Procedure

Depending on the composition of your deployment, there are three methods that you can use to get your data into i2 Analyze for analysis.

1. To develop the process for ingesting data into the Information Store, refer to Ingesting data into the Information Store.
2. To develop connections to external data sources, refer to Connecting to external data sources.
3. Ensure that analysts can import and create representative records in Analyst's Notebook. Then, if required, upload records to the Information Store. For more information, see Import data and Create i2 Analyze chart items.

When you develop the process to get your data into i2 Analyze, you might realize that your schema or security schema are not correct for your data. You can update the deployed schemas to better represent your data and security model. Some changes require you to remove and recreate the underlying database.

1. To update your deployed Information Store or Chart Store schema, refer to Changing the schema.
2. To update your deployed security schema, refer to Configuring the security schema.

After you develop the mechanisms for making data available to analysts, you can configure how analysts interact with the data when they use the system. The list of things that you can configure includes:

1. To configure which features or types of commands analysts can access, refer to Controlling access to features.
2. To configure how analysts search for information, and the options that are available to them, refer to Configuring search.
3. To configure how analysts can identify matching records, refer to Configuring matching.
4. To configure user security, refer to Configure user authentication and authorization.

   For more information about the configuration changes that you can make, see Configuring i2 Analyze.

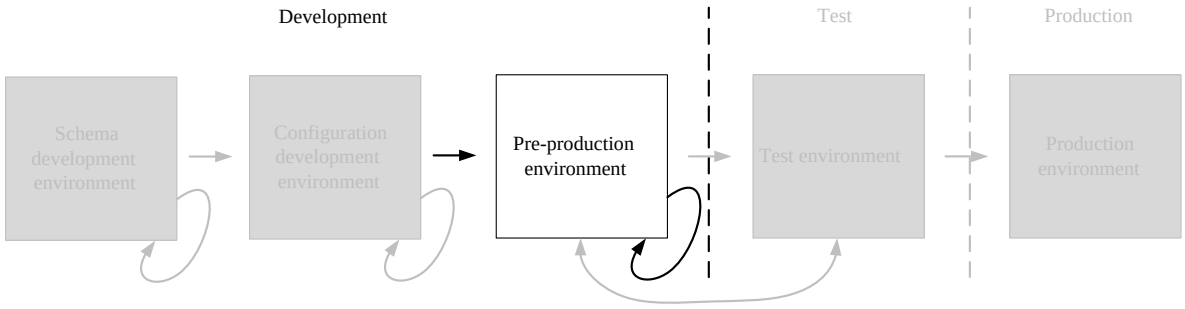
What to do next

After you configure your deployment sufficiently in the single-server environment, you can move to another environment that is more representative of the production deployment. Keep your configuration development environment in place so that you can access the `configuration` directory in later phases of the production process, and return to it to make further configuration changes.

Next, create the Pre-production environment.

Pre-production environment

Your target deployment topology is probably different from the single-server configuration development environment. In the pre-production environment, deploy i2 Analyze in the deployment topology that matches your intended production deployment.



After you develop the i2 Analyze configuration, you can use a more representative *pre-production environment* for aspects of the configuration that rely on environment-specific variables. In this environment, deploy i2 Analyze in the same physical deployment topology as the target production deployment. For example, you might modify the configuration to deploy the components of i2 Analyze on multiple servers or with high availability.

Creating the pre-production environment

In the pre-production environment, you can deploy i2 Analyze in the same physical deployment topology as your target production deployment. The process starts with your configuration from the configuration development environment, which you modify to match your chosen deployment topology.

Before you begin

Ensure that you have access to the following files and directories so that you can copy them to the pre-production development environment:

- The `toolkit\configuration` directory from the configuration development environment
- Liberty user configuration, for example:
  - `user-registry.xml`
  - `server.xml`
- Any configuration that you completed in the Information Store database. For example, your merged property values definition views.
- Any certificates and certificate stores that are required

If you are planning to deploy in production with high availability, you should configure and deploy high availability in the pre-production environment to develop the configuration. To create a pre-production environment configured for high availability, complete the instructions in i2 Analyze with high availability rather than the steps on this page.

About this task

Create your pre-production environment, and update your configuration to match the physical deployment topology of the pre-production environment.

Procedure

1. Install any prerequisite software to prepare your servers for the pre-production environment. For the pre-production environment, use the same deployment topology as your intended production environment. For more information, see Deployment topologies.
2. Copy the `toolkit\configuration` directory from the configuration development environment, to the `toolkit` directory at the root of the deployment toolkit on the Liberty server in the pre-production environment.
3. Update the values for any configuration settings that are specific to the environment.
   a. If you are creating a deployment with a database that is remote from the Liberty server, follow the instructions in Specifying remote database storage to update the `environment.properties` and `topology.xml` files for this deployment topology.
   b. If you are creating a deployment with multiple Solr and ZooKeeper servers, follow the instructions in Specifying remote Solr and ZooKeeper servers to update the `topology.xml` file for these deployment topologies.

   The `environment.properties`, `http-server.properties`, and `topology.xml` contain host name and file path settings that you might need to update for the servers in your pre-production environment. For more information, see Configuration files reference.
4. If your deployment uses a proxy server or load balancer to route requests from clients, ensure that it is configured for use with i2 Analyze. Specify the URI that clients use to connect to i2 Analyze. For more information, see Deploying a proxy server for use with i2 Analyze.

5. Deploy and start i2 Analyze:
   - In a single-server topology, or with a remote database only, see Deploying i2 Analyze.
   - In a multiple-server topology, see Deploying i2 Analyze on multiple servers.

After you deploy i2 Analyze, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze.

1. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment. This might involve copying the user registry file, or updating the `server.xml` file.
2. Complete any configuration changes in the Information Store database.
   a. If you created any rules or schedules to delete records by rule, replicate the rules and schedules that you created in the previous environment.
   b. If you created any merged property values definition views for your ingestion process, replicate the view definition that you created in the previous environment.

What to do next

After you deploy i2 Analyze in the pre-production development environment, you might want to configure aspects of the deployment that are topology specific. For example, in a multiple-server environment you can secure the connections between servers by using SSL.

To configure i2 Analyze in pre-production, see Configuring i2 Analyze in pre-production.

Configuring i2 Analyze in pre-production

In the pre-production environment, configure the aspects of i2 Analyze that depend on the deployment topology.

Procedure

1. To configure SSL connections between the servers in your deployment, refer to Secure Sockets Layer connections with i2 Analyze.
2. To develop the process for backing up and restoring your deployment of i2 Analyze, refer to Backing up a deployment.
3. To understand how to administer high availability and develop your continuous operation processes, refer to Administering high availability.

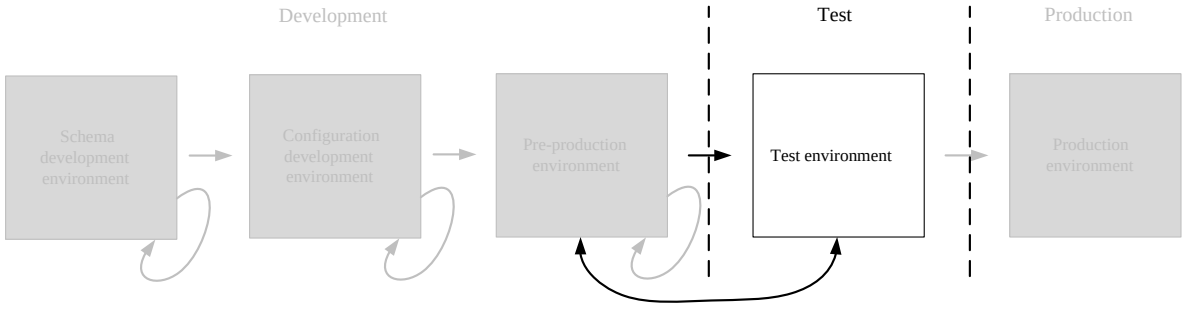
What to do next

After you configure i2 Analyze sufficiently, you can move to the test environment. Keep your pre-production environment in place so that you can access the `configuration` directory in later phases of the production process, and return to it to make further configuration changes.

Next, create the Test environment.

Test environment

The second phase of creating a production deployment focuses on testing. This phase is important, because it enables you to identify any changes to the environment or configuration that must be completed before you deploy into production.



You use a *test environment* to test the deployment to ensure that it meets the requirements of the production deployment. The test environment should match the production environment as closely as possible. In the test environment, perform comprehensive testing of the deployment against the final requirements with a selection of the users and a sample data set.

Creating the test environment

In the test environment, you deploy i2 Analyze with the configuration that you developed previously. With i2 Analyze deployed in the test environment, you can perform comprehensive testing of the deployment to confirm that it meets your production requirements.

Before you begin

Ensure that you have access to the following files and directories so that you can copy them to the test environment:

- The `toolkit\configuration` directory from the pre-production environment
- Liberty user configuration, for example the `server.xml` file
- Any configuration that you completed in the Information Store database. For example, your merged property values definition views.
- Any certificates and certificate stores that are required

If you are planning to deploy in production with high availability, complete the instructions in i2 Analyze with high availability rather than the steps on this page.

Procedure

1. Install any prerequisite software to prepare your servers for the test environment. For the test environment, use the same deployment topology as in your pre-production environment. For more information, see Deployment topologies.
2. Copy the `toolkit\configuration` directory from the pre-production environment to the `toolkit` directory at the root of the deployment toolkit on the Liberty server in the test environment.

3. Update the values for any configuration settings that are specific to the environment. `environment.properties`, `http-server.properties`, and `topology.xml` contain settings that you might need to update. For more information, see Configuration files reference.
4. If your deployment uses a proxy server or load balancer to route requests from clients, ensure that it is configured for use with i2 Analyze. Specify the URI that clients use to connect to i2 Analyze. For more information, see Deploying a proxy server for use with i2 Analyze.
5. Deploy and start i2 Analyze:
   - In a single-server topology, or with a remote database only, see Deploying i2 Analyze.
   - In a multiple-server topology, see Deploying i2 Analyze on multiple servers.

After you deploy, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze.

1. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment. This might involve copying the user registry file, or updating the `server.xml` file.
2. Complete any configuration changes in the Information Store database.
   a. If you created any rules or schedules to delete records by rule, replicate the rules and schedules in the current environment.
   b. If you created any merged property values definition views for your ingestion process, replicate the view definitions in the current environment.

What to do next

After you deploy i2 Analyze in the test environment, you can test your deployment to ensure that it meets the requirements of your organization.

For information about testing i2 Analyze, see Testing your deployment.

Testing your deployment

In your test environment, complete the comprehensive testing of the system that is required to prove that the deployment is ready for production.

About this task

Create a test charter that contains a list of tests that you can use to verify that your deployment is ready for production.

In your test environment, you should use real data at a representative volume.

Procedure

1. Create the test charter.

   Focus your test charter in the following areas that are applicable to your deployment:

   - Data ingestion into the Information Store
   - Data deletion from the Information Store
   - Data acquisition from external sources
   - Searching and analyzing data
   - Storing and retrieving charts in the Chart Store

- Performance of the system
- Backup and recovery of the system
- High availability and continuous operations after server failure

After you create a test charter for your deployment, you can start your testing.

1. Complete each test in your charter, and record if the deployment passed or failed each test.
2. If any of the tests fail, return to your development or pre-production environment and change the configuration. Then, deploy the same change to the test environment and repeat the testing.

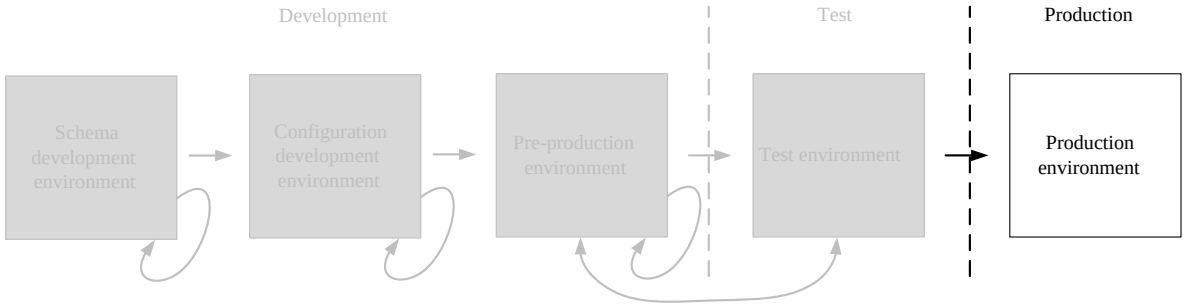
What to do next

After you test i2 Analyze sufficiently, you can move to the production phase. Keep your test environment in place, so that you can access the `configuration` directory in later phases of the production process and continue to test any changes to your configuration.

Next, complete the instructions in Production environment.

Production environment

The third phase of the deployment process focuses on deploying into production. In this phase you make your deployment of i2 Analyze available to users.



You use a *production environment* to host the deployment in production. When you have a configuration of i2 Analyze that passed your test phase, you can deploy i2 Analyze with that configuration into your production environment.

In the production environment, run the tests again to confirm that the deployment is working successfully. If these tests are successful, you can make the deployment available to users. If the tests are not successful, you must return to your test or development environment to make any necessary changes. Then, complete the test phase once more.

Creating the production environment

To deploy i2 Analyze in production, you use the i2 Analyze configuration that you developed previously. Before you deploy i2 Analyze, update the configuration to reflect any environment changes.

Before you begin

Ensure that you have access to the following files and directories so that you can copy them to the production environment:

- The `toolkit\configuration` directory from the test environment
- Liberty user configuration, for example the `server.xml` file
- Any configuration that you completed in the Information Store database. For example, your merged property values definition views.
- Any certificates and certificate stores that are required

If you are planning to deploy with high availability, complete the instructions in i2 Analyze with high availability rather than the steps on this page.

Procedure

1. Install any prerequisite software to prepare your servers for the production environment. For the production environment, use the same deployment topology as in your test environment. For more information, see Deployment topologies.
2. Copy the `toolkit\configuration` directory from the test environment to the `toolkit` directory at the root of the deployment toolkit on the Liberty server in the production environment.
3. Update the values for any configuration settings that are specific to the environment. `environment.properties`, `http-server.properties`, and `topology.xml` contain settings that you might need to update. For more information, see Configuration files reference.
4. If your deployment uses a proxy server or load balancer to route requests from clients, ensure that it is configured for use with i2 Analyze. Specify the URI that clients use to connect to i2 Analyze. For more information, see Deploying a proxy server for use with i2 Analyze.
5. Deploy and start i2 Analyze:
   - In a single-server topology, or with a remote database only, see Deploying i2 Analyze.
   - In a multiple-server topology, see Deploying i2 Analyze on multiple servers.

After you deploy i2 Analyze, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze.

1. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment. This might involve copying the user registry file, or updating the `server.xml` file.
2. Complete any configuration changes in the Information Store database.
   a. If you created any rules or schedules to delete records by rule, replicate the rules and schedules in the current environment.
   b. If you created any merged property values definition views for your ingestion process, replicate the view definitions in the current environment.

What to do next

Complete testing of the deployment to ensure that it is working in the production environment before you make i2 Analyze available to users.

Deployment resources

The deployment resources section contains information that is referenced elsewhere in the deployment section. The information is used in a many places throughout the deployment process.

Deployment topologies

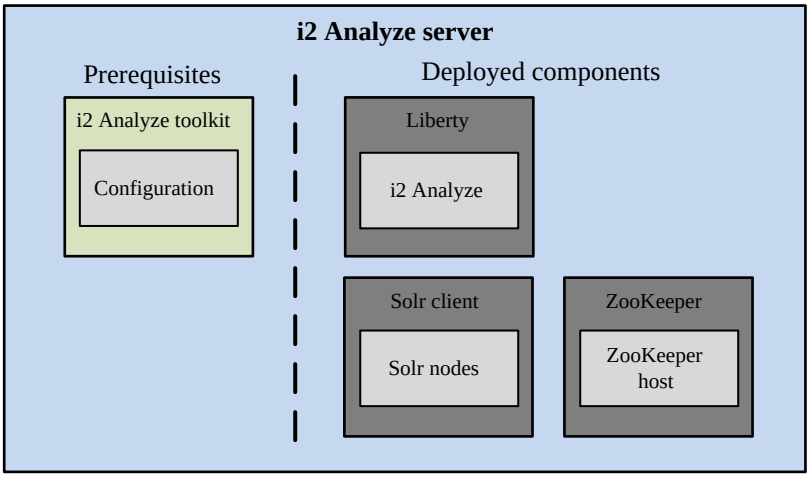
You can use the deployment toolkit to deploy i2 Analyze in a number of different physical topologies. Throughout the process of creating a production deployment, you might use several of them depending on your purpose at the time.

The following diagrams show the servers that are used in each deployment topology, the prerequisites that are required on each server, and the components of i2 Analyze that are deployed.

i2 Connect gateway only (single server)

In the i2 Connect gateway only deployment, all of the components of i2 Analyze are on the same server. No database management system is required.

i2 Connect only topology



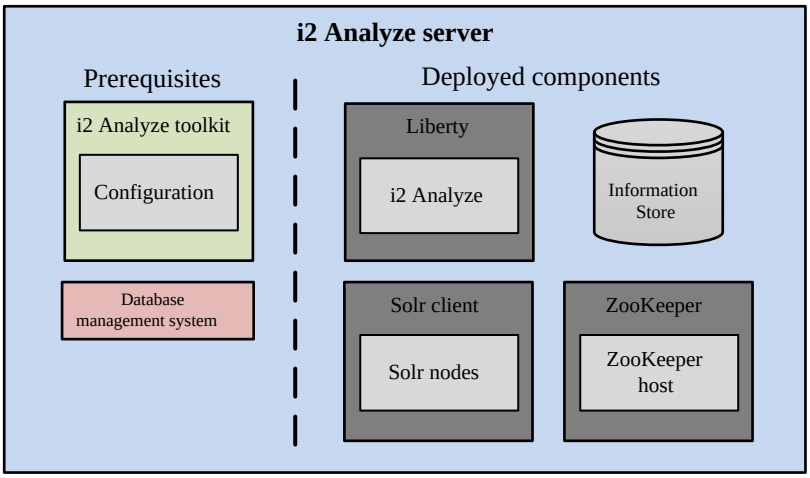
On the i2 Analyze server, install the prerequisites by following these instructions:

- Installing i2 Analyze

Single server

In the single-server topology, all of the components of i2 Analyze are on the same server.

Single-server topology



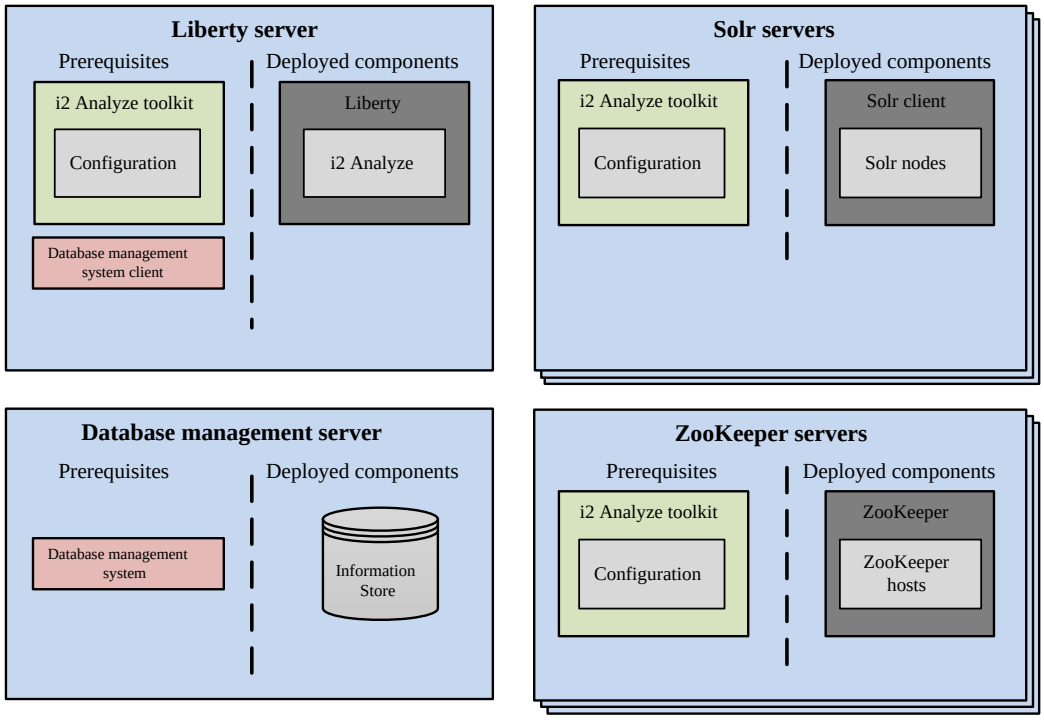
On the i2 Analyze server, install the prerequisites by following these instructions:

- Installing i2 Analyze
- Installing a database

Multiple servers

In the multiple-server topology, each component of i2 Analyze is on its own server.

Multiple-server topology



In the diagram, Liberty, Solr, ZooKeeper, and the Information Store (or the Chart Store) are each deployed on their own server.

You can also have a multiple-server deployment topology in which any of number of the components are located on the same server.

On the Liberty server, install i2 Analyze, the database management system client, and optionally the HTTP server:

- Installing i2 Analyze
- PostgreSQL client or Db2 client or SQL Server client

On your database management server, install your database management system:

- Installing PostgreSQL or Installing Db2 or Installing SQL Server

On each Solr and ZooKeeper server, install i2 Analyze:

- Installing i2 Analyze

The i2 Analyze deployment toolkit

The deployment toolkit contains scripts and components that you need to deploy and maintain i2 Analyze. Configuring components within the toolkit allows you to deploy consistently to the same locations with the same settings.

Directories

The deployment toolkit contains files in several directories. On most occasions, you need to interact with only three of the directories:

- The `examples` directory includes the base configurations that you can use to create example deployments, example data, and an example i2 Connect connector.
- The `configuration` directory contains files that you must update with information specific to your deployment.

  Note: When the deployment toolkit is first installed, this directory does not exist. At the start of deployment, you use one of the base configurations to create this directory.
- The `scripts` directory contains the `setup` script that you use to deploy and configure i2 Analyze.

The `setup` script

The `setup` script completes tasks that apply configuration and other changes to deployments of i2 Analyze. For a list of available tasks and other information, refer to Deployment toolkit tasks.

Alternatively, on the i2 Analyze server, open a command prompt and navigate to `toolkit\scripts` and run one of the following commands:

`setup -h`

The `-h` argument displays the usage, common tasks, and examples of use for the `setup` script.

`setup -a`

The `-a` argument displays the same content as when you use `-h`, and a list of additional tasks.

Note: On Linux, whenever you run the `setup` script you must prefix its name with '`./`'. For example, `./setup -t deploy`.

The base configurations

In the i2 Analyze deployment toolkit, several base configurations are available. To reduce the amount of configuration that you need to complete, you can start from the configuration that is closest to the requirements of your final deployment pattern.

The i2 Analyze deployment toolkit includes the following base configurations:

chart-storage

The `chart-storage` base configuration contains settings for a deployment of i2 Analyze that includes:

- A Chart Store only

Users can connect to the deployment with the following client:

- Analyst's Notebook

information-store-opal

The `information-store-opal` base configuration contains settings for a deployment of i2 Analyze that includes:

- An Information Store (and a Chart Store)

Users can connect to the deployment with the following clients:

- Analyst's Notebook
- i2 Notebook web client
- i2 Investigate web client

daod-opal

The `daod-opal` base configuration contains settings for a deployment of i2 Analyze that includes:

- The i2 Connect gateway only, which can provide access to other data stores through connectors that you create

Users can connect to the deployment with the following client:

- Analyst's Notebook

chart-storage-daod

The `chart-storage-daod` base configuration contains settings for a deployment of i2 Analyze that includes:

- A Chart Store
- The i2 Connect gateway, which can provide access to other data stores through connectors that you create

Users can connect to the deployment with the following clients:

- Analyst's Notebook
- i2 Notebook web client

information-store-daod-opal

The `information-store-daod-opal` base configuration contains settings for a deployment of i2 Analyze that includes:

- An Information Store (and a Chart Store)
- The i2 Connect gateway, which can provide access to other data stores through connectors that you create

Users can connect to the deployment with the following clients:

- Analyst's Notebook
- i2 Notebook web client
- i2 Investigate web client

It is possible to extend your deployment after you select and deploy a base configuration by adding the appropriate components and configuration files.

For more information about the components that are included in a deployment of i2 Analyze, see Understanding components.

Example schemas

The i2 Analyze deployment toolkit includes example schema files that you can use as a starting point for your own schemas.

Example i2 Analyze schemas

The deployment toolkit includes five pairs of example i2 Analyze schemas and associated charting schemes in the `toolkit\examples\schemas` directory:

Law Enforcement

The law enforcement schema deals with criminal activity. It contains entity and link types that are designed to track connections within criminal networks.

Commercial Insurance

The commercial insurance schema deals with fraud in a commercial setting. It contains entity and link types that are designed to track financial transactions such as credit card payments and insurance claims.

Military

The military schema helps with military intelligence tracking. It contains entity and link types that target military operations.

Signals Intelligence

The signals intelligence schema focuses particularly on the cellphones and cell towers that are involved in mobile telecommunications, and on the calls that take place between them.

Chart Storage

The chart storage schema is not an example of a particular domain, but rather the starting point for any deployment of i2 Analyze that includes only the Chart Store.

Example security schema

The example security schema files specify a number of security dimensions and dimension values, and security groups for users.

After you create the `toolkit\configuration` directory by copying it from the base configuration of your choice, you can find an appropriate example security schema file in the `toolkit\configuration\examples\security-schema` directory.

Deployment toolkit tasks

The following toolkit options and tasks are available to use with the `setup` script in the deployment toolkit.

usage: setup -t TASK

Argument name	Argument Description
`-h, --help`	Shows this help message, and exits
`-a, --additional`	Shows additional arguments and tasks, and exits

Argument name	Argument Description
`-t, --task <task>`	Specifies the task to perform
`-s, --server <server>`	Specifies the server profile to manage
`-w, --war <war>`	Specifies the war name
`-co, --collection <collection>`	Specifies the solr collection ID
`-id, --id <ids>`	Specifies the ids of the components to manage
`-hn, --hostname <hostname>`	Restrict the task to operate only on components with the specified hostname attribute
`-l, --locale <locale>`	Specify the language code of the schema files to use in the example deployment
`--all`	Apply the operation to all applicable components
`-sn, --schemaName <Schema>`	Specify the name of the schema file to use in the example deployment
`-e, --exampleData <Example Data>`	Specify the name of the directory containing the example data to ingest
`-f, --force`	Suppress the warning prompt that is associated with tasks that might result in data loss
`--scripts`	Generates scripts
`-st, --stacktrace`	Print the full stacktrace if an error occurs

The following installation and deployment tasks are available

Name	Description
`installLiberty`	Installs Liberty from the Liberty binaries in the toolkit, to the directory specified in environment.properties.
`installZookeeper`	Installs ZooKeeper from the ZooKeeper binaries in the toolkit, to the directory specified in environment.properties.
`installSolr`	Installs Solr from the Solr binaries in the toolkit, to the directory specified in environment.properties.
`deployExample`	Deploys i2 Analyze with default files and settings.

Name	Description
`ingestExampleData`	Ingests entity and link record examples into the Information Store.
`deploy`	Creates the databases, creates the application profile, and deploys i2 Analyze.
`deployLiberty`	Deploys the i2 Analyze application into Liberty.
`start`	Starts all i2 Analyze services on the current server.
`stop`	Stops all i2 Analyze services on the current server.
`restart`	Restarts all i2 Analyze services on the current server.
`configSummary`	Summarizes the configuration of the toolkit.
`version`	Summarizes the deployment version information.

Examples of use:

- `setup -t deployExample`
- `setup -t ingestExampleData`
- `setup -t deploy`
- `setup -t start`
- `setup -t configSummary`

The following upgrade tasks are available

Name	Description
`upgrade`	Upgrades the configuration files, and upgrades i2 Analyze.
`upgradeConfiguration`	Upgrades the configuration files for an Information Store deployment.
`upgradeZookeeper`	Upgrades ZooKeeper and ZooKeeper configuration files to the version required by the toolkit.
`upgradeSolr`	Upgrades Solr and Solr configuration files to the version required by the toolkit.

Name	Description
`upgradeDatabases`	Upgrades the Information Store database and clears the search index if required.
`upgradeSolrCollections`	Upgrades ZooKeeper and Solr, and creates a new collection if required.
`upgradeLiberty`	Upgrades Liberty to the version required by the toolkit.

Examples of use:

- `setup -t upgrade`
- `setup -t upgradeConfiguration`
- `setup -t upgradeSolr -hn "example.solr.hostname"`

The "upgradeZookeeper", "upgradeSolr", "upgradeDatabases", and "upgradeSolrCollections" tasks support an optional -hn argument that restricts their effect to a single host.

The following administration tasks are available

Name	Description
`replayFromTimestamp`	Starts Liberty in a mode that replays all events since the time specified by the 'datetime.to.replay.from' property in environment-advanced.properties.
`configureHttpServer`	(Deprecated) Sets up the reverse proxy configuration for IBM HTTP Server.
`enableLibertyAdminCenter`	Enable the Liberty Admin Center.
`disableLibertyAdminCenter`	Disable the Liberty Admin Center.
`generateDefaults`	Configures the environment with default property values.
`ensureBasicUserRegistry`	Configures the application for basic user registry authentication.
`ensureExampleUserRegistry`	Populates the user registry with an example user and user groups that map to the example security schema.
`addInformationStore`	Generates a fragment for the Information Store, and updates topology.xml.
`addI2Connect`	Updates topology.xml for i2 Connect.

Name	Description
`updateConnectorsConfiguration`	Updates the i2 Analyze server with the connection details of the connectors defined in topology.xml. In a deployment without the Information Store, the i2 Analyze schema and charting schemes are also updated.
`updateSchema`	Updates i2 Analyze to conform to the schema file referenced in the configuration.
`updateSecuritySchema`	Updates i2 Analyze to use the security schema file referenced in the configuration.
`updateLiveConfiguration`	Updates the i2 Analyze server with the latest version of the files in the configuration/live directory from the deployment toolkit.
`generateAnalyzeSchemaFromIBase`	Generates an i2 Analyze schema and charting schemes from an iBase database.
`createDatabaseStorage`	Creates the database storage*
`createDatabases`	Creates the database storage and tables*
`modifyInformationStoreDatabase`	Runs the informationStoreModifications.sql script on the Information Store database.
`dropTables`	Drops all of the tables from the database but leaves the database intact*
`dropDatabases`	Drops the entire database and de-allocates storage*
`emptyInformationStore`	Empties the Information Store of data, apart from metadata.
`addInformationStoreIngestionSource`	Adds or replaces information about an ingestion source to the Information Store.
`createInformationStoreStagingTable`	Creates an Information Store staging table for a specific entity or link type.
`ingestInformationStoreRecords`	Ingests records into the Information Store.
`deleteProvenance`	Deletes (entity/link) provenance from the Information Store.
`previewDeleteProvenance`	Previews deleting (entity/link) provenance from the Information Store.

Name	Description
`syncInformationStoreCorrelation`	Synchronizes data in the Information Store after a correlation operation failed during ingestion.
`enableMergedPropertyValues`	Creates the database views used to define the property values of merged i2 Analyze records.
`disableMergedPropertyValues`	Removes the database views used to define the property values of merged i2 Analyze records.
`duplicateProvenanceCheck`	Checks the Information Store for duplicated origin identifiers. Any provenance that has a duplicated origin identifier is added to a staging table.
`duplicateProvenanceDelete`	Deletes (entity/link) provenance from the Information Store that has duplicated origin identifiers. The provenance to delete is identified in the staging tables created by the duplicateProvenanceCheck task.
`deleteOrphanedDatabaseObjects`	Deletes (entity/link) database objects that are not associated with an i2 Analyze record from the Information Store.
`createEtlToolkit`	Creates a DataStage ETL toolkit that contains the files DataStage requires to run pipeline jobs.
`generateInformationStoreIndexCreationScripts`	Generates InfoStore 'create indexes' DDL scripts for the specified item type
`generateInformationStoreIndexDropScripts`	Generates InfoStore 'drop indexes' DDL scripts for the specified item type
`clearData`	Clears the search index and all the data in the database.
`clearSearchIndex`	Clears the search index.
`clearLTPAkeys`	Clears the LTPA keys.
`clearInformationStoreStagingSchema`	Clears all the tables in the Information Store Staging Schema.
`dropInformationStoreErrorTables`	Removes the _ERROR and _REJECT tables from the Information Store.
`backupDatabases`	Backs up the database. i2 Analyze must be stopped first.

Name	Description
`restoreDatabases`	Restores the database from a specified timestamp. i2 Analyze must be stopped first.
`backupSolr`	Backs up the Solr index and ZooKeeper configuration.
`restoreSolr`	Restores the Solr index and ZooKeeper configuration from a specified timestamp. i2 Analyze must be stopped first.
`backupConfiguration`	Backs up the i2 Analyze and Liberty configuration.
`restoreConfiguration`	Restores the i2 Analyze and Liberty configuration from a specified timestamp. i2 Analyze must be stopped first.
`validateBackups`	Validates that the timestamps of the backups specified are in the correct chronological order.
`resetPrivacyAgreements`	Resets privacy agreements acceptance state.
`showChanges`	Outputs the changes to i2 Analyze since the previously installed version to the command line.

The "clearData" and "clearSearchIndex" tasks support an optional -co argument that restricts their effect to a single Solr collection.

Example of use:

- `setup -t clearData -co "solr.collection.id"`

* The exact behavior of these tasks may change depending on the chosen database engine.

Tasks for DB2 only

Name	Description
`catalogRemoteDB2Nodes`	Adds a remote database server entry to the DB2 node directory for each remote DB2 database that is defined in topology.xml
`uncatalogRemoteDB2Nodes`	Removes the remote database server entry in the DB2 node directory for each remote DB2 database that is defined in topology.xml.
`recatalogRemoteDB2Nodes`	Removes, then re-adds the remote database entry in the DB2 node directory for each remote DB2 database that is defined in topology.xml.

Name	Description
`listDB2NodeDirectory`	Lists the contents of the DB2 node directory.
`catalogDB2Databases`	Adds an entry to the system database directory for each DB2 database that is defined in topology.xml. If the database is remote from the i2 Analyze server, the database is cataloged at the node specified for that database in topology.xml.
`uncatalogDB2Databases`	Removes the entry in the system database directory for each DB2 database that is defined in topology.xml.
`recatalogDB2Databases`	Removes, then re-adds the entry in the system database directory for each DB2 database that is defined in topology.xml.
`listDB2SystemDatabaseDirectory`	Lists the contents of the local DB2 system database directory.

Examples of use:

- `setup -t catalogRemoteDB2Nodes`
- `setup -t catalogDB2Databases`

In addition to the start, stop and restart tasks, the following tasks are available

Name	Description
`startLiberty`	Starts Liberty.
`stopLiberty`	Stops Liberty.
`restartLiberty`	Restarts Liberty.
`startSolrAndZk`	Starts the Solr nodes and ZooKeeper hosts.
`stopSolrAndZk`	Stops the Solr nodes and ZooKeeper hosts.
`restartSolrAndZk`	Restarts the Solr nodes and ZooKeeper hosts.
`startSolrNodes`	Starts the Solr nodes.
`stopSolrNodes`	Stops the Solr nodes.
`restartSolrNodes`	Restarts the Solr nodes.
`startZkHosts`	Starts ZooKeeper hosts.

Name	Description
`stopZkHosts`	Stops ZooKeeper hosts.
`restartZkHosts`	Restarts ZooKeeper hosts.

Examples of use:

- `setup -t startSolrAndZk`
- `setup -t stopZkHosts -id "1,3"`
- `setup -t restartSolrNodes -id node1`
- `setup -t startSolrNodes -hn "example.solr.hostname"`

The "SolrNodes" and "ZkHosts" tasks support an optional -id argument.

The comma-separated list of identifiers that you specify restricts the task to the nodes and hosts with matching identifiers in the topology.

The "SolrNodes" and "ZkHosts" tasks support an optional -hn argument that restricts their effect to a single host.

The following Solr and ZooKeeper tasks are available

Name	Description
`createSolrNodes`	Creates the Solr nodes that are defined in topology.xml. If the nodes already exist, their configuration is updated.
`createZkHosts`	Creates the ZooKeeper hosts that are defined in topology.xml. If the hosts already exist, their configuration is updated.
`getZkStatus`	Reports the status of the ZooKeeper hosts that are defined in topology.xml.
`createAndUploadSolrConfig`	Creates and uploads the Solr configuration to the ZooKeeper hosts.
`createSolrCollections`	Creates the Solr collections that are defined in topology.xml.
`deleteSolrCollections`	Deletes the Solr collections that are defined in topology.xml.
`addSolrReplicas`	Creates more Solr replicas after the num-replicas setting in topology.xml is increased.
`deleteExcessSolrReplicas`	Deletes Solr replicas after the num-replicas setting in topology.xml is decreased.

Name	Description
`deleteAllSolrReplicas`	Deletes Solr replicas until there is only one replica per shard.
`solrReplicaStatus`	Checks that all Solr replicas are in sync with each other.
`updateSolrReplicaPlacementPlugin`	Updates the Solr replica placement plugin with the configuration from solr.replica.placement.plugin.json.
`checkSolrCollectionVersions`	Checks the versions of all Solr collections and reports on whether they need to be upgraded. Use the -co flag to check a specific collection.
`upgradeAllSolrIndexes`	Deletes and re-creates the Solr collections to bring them up to date. As a result, the data for the 'main', 'match', and 'chart' indexes is reindexed. Other index types are cleared.
`upgradeMainIndex`	Deletes and re-creates the 'main' Solr collection to bring it up to date. As a result, the data is reindexed.
`upgradeMatchIndex`	Deletes and re-creates the 'match' Solr collection to bring it up to date. As a result, the data is reindexed.
`upgradeChartIndex`	Deletes and re-creates the 'chart' Solr collection to bring it up to date. As a result, the data is reindexed.
`upgradeTransientIndexes`	Deletes and re-creates the transient Solr collections to bring them up to date. As a result, the data is cleared.

Examples of use:

- `setup -t createSolrNodes -hn "example.solr.hostname"`
- `setup -t createSolrCollections -co "solr.collection.id"`

These tasks support an optional -hn argument that restricts their effect to a single host.

The "createAndUploadSolrConfig", "createSolrCollections", and "deleteSolrCollections" tasks support an optional -co argument that restricts their effect to a single Solr collection.

The following tasks manage main indexes and match indexes. ZooKeeper and the application server must be running for these commands to succeed

Name	Description
`switchStandbyMatchIndexToLive`	Sets the standby match index to live if it is READY, and sets the previous live match index to standby.
`clearStandbyMatchIndex`	Deletes the contents of the standby match index and sets its state to DISABLED.
`updateMatchRules`	Uploads the system match rules and applies them to the standby match index, which starts BUILDING again.
`switchStandbyMainIndexToLive`	Sets the standby main index to live if it is READY, and sets the previous live main index to standby.
`clearStandbyMainIndex`	Deletes the contents of the standby main index and sets its state to DISABLED.
`rebuildStandbyMainIndex`	Instructs the standby main index to start BUILDING again, if it exists.
`switchStandbyChartIndexToLive`	Sets the standby chart index to live if it is READY, and sets the previous live chart index to standby.
`clearStandbyChartIndex`	Deletes the contents of the standby chart index and sets its state to DISABLED.
`rebuildStandbyChartIndex`	Instructs the standby chart index to start BUILDING again, if it exists.
`pauseIndexPopulation`	Pauses index population for a specified index or all indexes. Population is paused until the Liberty server restarts.
`resumeIndexPopulation`	Resumes index population for a specified index or all indexes where population is currently paused.

i2 Analyze deployment settings

Some settings in the deployment toolkit require values that are specific to the environment that you are deploying in. Although these settings can sometimes use the default values, at different times during

the production process you might need to specify values that meet your environment and deployment requirements.

Specifying the deployment credentials

To allow the deployment toolkit to update the database, Lightweight Third-Party Authentication (LTPA) keys, and Solr search platform components, you must provide user names and passwords. The user names and passwords that you provide allow the system to set up and administer components of i2 Analyze, and are not used to access i2 Analyze.

About this task

Database

For the database that is identified in `topology.xml`, you must specify a user name and a password in the `credentials.properties` file. The `setup` script uses this information to authenticate with the database.

Note: The user that you specify must have privileges to create and populate databases in the database management system.

The database credentials are stored in the following format:

```
db.infostore.user-name=<user name>
db.infostore.password=<password>
```

For example:

```
db.infostore.user-name=admin
db.infostore.password=password
```

The `db.infostore.truststore.password` credential is used only when you configure the connection between the database and Liberty to use SSL. If you are not using SSL to secure this connection, you do not need to specify a value for the `db.infostore.truststore.password` credential. For more information about configuring SSL, see Configure Secure Sockets Layer with i2 Analyze.

If you change this password after deployment, the change will take effect when you redeploy the system.

LTPA keys

You must provide a value for the `ltpakeys.password` property. This value is used by the system to encrypt LTPA tokens.

- For a stand-alone deployment of i2 Analyze, you can specify any value as the password.
- For a deployment of i2 Analyze that uses LTPA tokens to authenticate with other systems, you must specify the same password that those systems use.

If you change this password after deployment, you must run the `setup -t clearLTPAkeys` toolkit task before the change will take effect when you redeploy the system.

Solr search platform

The Solr search platform is used to search data in the Information Store. You must provide values for the `solr.user-name` and `solr.password` properties. Any Solr indexes are created when i2 Analyze is first deployed, and the values that you provide here become the Solr user name and password.

If you have already deployed i2 Analyze, and you want to change the Solr password, new properties must be created in the following format:

```
solr.user-name.new=value
solr.password.new=value
```

For example:

```
solr.user-name=admin
solr.password={enc}E3FGHjYUI2A\=

solr.user-name.new=admin1
solr.password.new=password
```

After you provide the new values, run the `setup -t configureSolrSecurity` toolkit task. You must also run `setup -t deployLiberty` on your Liberty server, and `setup -t createSolrNodes` on your Solr servers. As a part of the deployment process, the new values replace the original values in the file.

When you deploy i2 Analyze, the passwords in the `credentials.properties` file are encoded.

Procedure

1. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
2. Specify the database, LTPA, and Solr credentials for your deployment:
   - the database user names and passwords
   - the LTPA keys password
   - the Solr user name and password
3. Save and close the `credentials.properties` file.

Specifying the JDBC driver

The application server requires a JDBC driver to enable communication with the database. The JDBC driver that you provide to the deployment toolkit depends on the database management system to use with the deployment.

Procedure

1. Locate the JDBC driver for your database management system.
   - If you are using PostgreSQL, download the latest driver from jdbc.postgresql.org.
   - If you are using Db2®, locate the `IBM\SQLLIB\java\db2jcc4.jar` file.
   - If you are using SQL Server, download the latest version of the Microsoft JDBC Driver for SQL Server that's compatible with your system from https://learn.microsoft.com/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server. Extract the contents of the download, and locate the file whose name matches the pattern `mssql-jdbc-x.y.z.jre11.jar`.
2. Copy the relevant driver file for your database management system to the `toolkit\configuration\environment\common\jdbc-drivers` directory.

   Note: Ensure that the user you use to run the deployment commands has write permissions for the JDBC jar file.

Specifying remote database storage

You can deploy i2 Analyze with storage that is remote from the i2 Analyze server. You can choose to configure either IBM Db2 or SQL Server database storage.

Configuring remote IBM Db2 database storage

You can deploy i2 Analyze with Db2 database storage that is remote from the i2 Analyze server. When i2 Analyze is configured to deploy with a remote instance of Db2, the database is created and updated remotely without the i2 Analyze deployment toolkit present on the server that hosts the Db2 instance.

Before you begin

To deploy an i2 Analyze server that uses remote Db2 database storage, you must install Db2 on your database server, and Db2 or IBM Data Server Client on the application server. Both instances of Db2 must be installed according to the specifications defined in Installing Db2 for i2 Analyze.

About this task

To deploy with remote storage, the i2 Analyze configuration must contain certain information about your Db2 instances. After you update the configuration, the specified databases can be created and updated by the deployment toolkit on the remote server.

Procedure

1. Edit the `configuration\environment\topology.xml` file to specify your remote Db2 databases:
   a. Update the `host-name` and `port-number` attribute values of the `<database>` element to match the values of your remote Db2 instance.
   b. Add the `node-name` and `os-type` attributes to the `<database>` element of the database to be hosted remotely.

      For example:

      ```
      <database database-type="InfoStore" dialect="db2" instance-name="DB2"
          database-name="ISTORE" xa="false" id="infostore"
          host-name="hostname" port-number="50000"
          node-name="node1" os-type="WIN" />
      ```

      The value for `node-name` is the name of the node to create in the Db2 node directory. The value of the `node-name` attribute must start with a letter, and have fewer than nine characters. For more information about naming in Db2, see Naming conventions.

      The value for `os-type` is the operating system of the remote Db2 server. The value of the `os-type` attribute must be one of the following values: `AIX`, `UNIX`, or `WIN`.

   Note: The value of the `instance-name` attribute must match the instance name of the remote Db2 instance.

   You can use the `db2level` command to get the name of your remote instance. For more information, see db2level - Show Db2 service level command.
2. Edit the `configuration\environment\opal-server\environment.properties` file, to specify the details of your remote and local instance of Db2.
   a. Ensure that the value of the `db.installation.dir` property is set for the local instance of Db2 or Data Server Client on the Liberty server.

If you are using a non-root installation, set the value for this property to the `sqllib` directory in the installation user's home directory. For example, `/home/db2admin/sqllib`.

b. Set the value of the `db.database.location.dir` property for the remote instance of Db2 on the database server.

3. Ensure that the users that are specified for your databases in the `configuration\environment\credentials.properties` file are valid for your remote instance of Db2 on the database server.
4. Catalog the remote node:

   ```
   setup -t catalogRemoteDB2Nodes
   ```

   You can complete the steps that are performed by the `catalogRemoteDB2Nodes` task manually. For example, if you are deploying a system that uses Transport Layer Security (TLS).

   To catalog the remote nodes manually, you can run the `CATALOG TCPIP NODE` instead of using the `setup -t catalogRemoteDB2Nodes` command. For more information about the command, see `CATALOG TCPIP/TCPIP4/TCPIP6 NODE` command.

   The following table shows how the `CATALOG` command parameters map to the values in the `topology.xml` file:

`CATALOG TCPIP NODE` command parameters	**`<database>` element attributes**
`TCPIP NODE` *`nodename`*	`node-name`
`REMOTE` *`hostname`*	`host-name`
`SERVER` *`port number`*	`port-number`
`REMOTE_INSTANCE` *`instance-name`*	`instance-name`

Results

When i2 Analyze is deployed, the Information Store database is created or updated on the remote database management system. A remote node is created with the name that is specified for the `node-name` attribute, and the database is cataloged against that node.

To check that the remote nodes and databases are cataloged, you can use the `listDB2NodeDirectory` and `listDB2SystemDatabaseDirectory` tasks after you deploy i2 Analyze:

- The `listDB2NodeDirectory` task lists the contents of the Db2 node directory.
- The `listDB2SystemDatabaseDirectory` task lists the contents of the local Db2 system database directory.

Configuring remote SQL Server database storage

You can deploy i2® Analyze with SQL Server database storage that is remote from the i2 Analyze server. When i2 Analyze is configured to deploy to a remote instance of SQL Server, the Information Store database can be created and updated remotely without the i2 Analyze deployment toolkit present on the server that hosts the SQL Server instance.

Before you begin

To deploy an i2 Analyze server that uses remote SQL Server database storage for the Information Store, you must install SQL Server on your database server, and the SQL Server client tools on your

application server. SQL Server and the client tools must be installed according to the specifications defined in Installing Microsoft SQL Server for i2 Analyze.

About this task

To deploy with remote storage, the deployment toolkit must contain certain information about your SQL Server installation. After you update the configuration, the specified database can be created and updated on the remote server by the deployment toolkit.

Procedure

1. Edit the `configuration\environment\topology.xml` file to specify the details of your remote database:
   a. Update the `host-name`, and either the `port-number` or `instance-name`, attribute values of the `<database>` element for the Information Store to match the values for SQL Server on your database server. If you are using SQL Server on Linux, you must use the `port-number` attribute.
   b. Add the `os-type` attribute to the `<database>` element for the Information Store database. The value of the `os-type` is used to support the search functions for the Information Store.

      For example:

      ```
      <database database-type="InfoStore" dialect="sqlserver"
        instance-name="<remote.instance.name>"
        database-name="ISTORE" xa="false" id="infostore"
        host-name="<remote.hostname>" os-type="WIN" />
      ```

      Where the value for `os-type` is the operating system of the remote database server. The value of the `os-type` attribute must be one of the following values: `UNIX` or `WIN`.

   For more information about the `<database>` element attributes, see Databases.
2. Edit the `configuration\environment\opal-server\environment.properties` file to specify the details of your remote and local installations of SQL Server:
   a. Ensure that the value of the `db.installation.dir` property is set for the local installation of SQL Server or the Microsoft™ Command Line Utilities for SQL Server on the Liberty server.
   b. Set the value of the `db.database.location.dir` property for the remote installation of SQL Server on the database server.
3. Ensure that the user that is specified for your database in the `configuration\environment\credentials.properties` file is valid for your remote installation of SQL Server on the database server.

Results

When i2 Analyze is deployed, the Information Store database is created or updated on the remote database management system.

What to do next

If the connection details for the remote database management system change, you can update the `topology.xml` file and redeploy the system.

Configuring remote PostgreSQL database storage

You can deploy i2 Analyze with PostgreSQL database storage that is remote from the i2 Analyze server. When i2 Analyze is configured to deploy with a remote instance of PostgreSQL, the database is created and updated remotely without the i2 Analyze deployment toolkit present on the server that hosts the PostgreSQL instance.

About this task

By default, PostgreSQL deployments support connections only from localhost. You must update the PostgreSQL configuration to allow external connections before you can configure i2 Analyze to connect to it.

To deploy with remote storage, the i2 Analyze configuration must contain certain information about your PostgreSQL instances. After you update the configuration, the specified databases can be created and updated by the deployment toolkit on the remote server.

Procedure

1. Enable PostgreSQL to support remote connections.
   a. In a text editor, open the `pg_hba.conf` file in the `data` folder on your PostgreSQL database server.
      1. To allow connection from any host to all PostgreSQL databases, using any PostgreSQL user and password, add these lines to the end of the `pg_hba.conf` file:

         ```
         host    all     all     0.0.0.0/0       scram-sha-256
         host    all     all     ::/0            scram-sha-256
         ```

         If you need to set restrictions, see the instructions at the top of the `pg_hba.conf` file and modify the file as required.
   b. In a text editor, open the `postgresql.conf` file in the `data` folder.
      1. Set the `listen_addresses` to `'*'` to allow connection from any host.
         If you need to set restrictions, see the comments for `listen_addresses` in the `postgresql.conf` file and modify the file as required.

         Note: When you allow connections from *any* host, this includes connections from localhost.
2. In the i2 Analyze configuration, edit the `configuration\environment\topology.xml` file. Update the `host-name` and `port-number` attribute values of the database element to match the values of your remote PostgreSQL instance, for example:

   ```
   <database database-type="InfoStore" dialect="postgres" database-
   name="ISTORE" xa="false" id="infostore" host-name="hostname" port-
   number="5432" os-type="WIN"/>
   ```
3. Edit the `configuration\environment\opal-server\environment.properties` file, to specify the details of your remote and local instance of PostgreSQL:
   - Ensure that the `db.installation.dir` property is set to the client 'Command Line Tools' installation folder.
   - Ensure that the `db.database.location.dir` property is set to the root data folder of the installation of PostgreSQL on the database server.
4. Ensure that the users that are specified for your databases in the `configuration\environment\credentials.properties` file are valid for your remote instance of PostgreSQL on the database server.

Deploying with PostgreSQL on Amazon Relational Database Service (Amazon RDS)

You can deploy i2 Analyze with a remote PostgreSQL database hosted on Amazon Relational Database Service (Amazon RDS).

Configuring RDS for i2 Analyze

1. To enable i2 Analyze to perform scheduled operations against the database, enable the `pg_cron` extension in your RDS instance. Follow the instructions to set up in the `pg_cron` extension and grant database users permissions described in Scheduling maintenance with the PostgreSQL pg_cron extension.
   a. By default, cron jobs run in the postgres database. For cron jobs to run in the ISTORE database, follow the instructions in Scheduling a cron job for a database other than the default database
2. By default, i2 Analyze requires the use of the `pg_default` tablespace. To enable i2 Analyze to use it, you must assign your master user as owner of the `pg_default` tablespace.

   For example:

   ```
   ALTER TABLESPACE pg_default OWNER TO postgres;
   ```

   Where `postgres` is the name of your master owner.

Configuring i2 Analyze for RDS

Follow the instructions in Configuring remote PostgreSQL database storage to configure i2 Analyze to connect to your RDS PostgreSQL database.

Deploying i2 Analyze

Continue to deploy i2 Analyze in the usual way. i2 Analyze deployment commands

Deploying with Microsoft SQL Server on Amazon Relational Database Service (Amazon RDS)

You can deploy i2 Analyze with a remote Microsoft SQL Server database hosted on Amazon Relational Database Service (Amazon RDS).

Configuring RDS for i2 Analyze

1. To enable i2 Analyze to perform scheduled operations against the database, SQL Server Agent is used. By default, only the master user is enrolled in the `SQLAgentUserRole` role. Enroll your database admin user to the role by completing the following instructions: Adding a user to the SQLAgentUser role.

Configuring i2 Analyze for RDS

Follow the instructions in Configuring remote SQL Server database storage to configure i2 Analyze to connect to your RDS SQL Server database.

Deploying i2 Analyze

Continue to deploy i2 Analyze in the usual way. i2 Analyze deployment commands

Specifying remote Solr and ZooKeeper servers

In a deployment of i2 Analyze, the location of each Solr server and ZooKeeper server is specified in the `topology.xml` file. When you run the commands to deploy i2 Analyze, the Solr nodes and ZooKeeper servers are created on servers that you specify.

About this task

Add the information about your Solr and ZooKeeper servers to the `configuration\environment\topology.xml` file.

Procedure

1. In the `topology.xml` file, add at least one `<solr-node>` element for each Solr server in your deployment topology.

   For example:

   ```
   <solr-nodes>
     ...
     <solr-node
       memory="2g"
       data-dir="C:/i2/i2analyze/data/solr"
       host-name="<solr_server_host_name>"
       id="node2"
       port-number="8984"
     />
   </solr-nodes>
   ```

   Where `<solr_server_host_name>` is the hostname of a Solr server.

   For more information about the possible values for each attribute, see Solr and ZooKeeper.

2. Edit the `configuration\environment\topology.xml` file to specify your ZooKeeper servers.

   For example:

   ```
   <zookeeper id="zoo">
     <zkhosts>
       <zkhost
         host-name="<zookeeper_server1_host_name>" id="1"
         port-number="9983" quorum-port-number="10483" leader-port-
   number="10983"
         data-dir="C:/i2/i2analyze/data/zookeeper"
         />
       <zkhost
         host-name="<zookeeper_server2_host_name>" id="2"
         port-number="9983" quorum-port-number="10483" leader-port-
   number="10983"
         data-dir="C:/i2/i2analyze/data/zookeeper"
         />
       <zkhost
         host-name="<zookeeper_server3_host_name>" id="3"
         port-number="9983" quorum-port-number="10483" leader-port-
   number="10983"
         data-dir="C:/i2/i2analyze/data/zookeeper"
         />
     </zkhosts>
   </zookeeper>
   ```

   Where `<zookeeper_serverx_host_name>` is the hostname of the ZooKeeper server.

Each <zkhost> element must have a unique value for the id attribute.

For more information about the possible values for each attribute, see Solr and ZooKeeper.

Specifying the Java distribution

i2 Analyze comes pre-packaged with the Adoptium Eclipse Temurin JDK 17 distribution of Java. You can change the distribution of Java that is used, and also you can specify different distributions for different components of i2 Analyze.

Before you begin

Any Java distribution that you specify for use with i2 Analyze and its components must be Java version 17.

Note: If you decide to use a different distribution, you assume responsibility for updating that distribution for security patches or if the version required by the i2 Analyze changes.

About this task

To change the Java distribution to use, you modify the `setup.in` script in the `configuration\environment` directory to contain the path where it is installed. If you are using the Linux distribution of the i2 Analyze toolkit, this is the `setup.in` file. On Windows it is `setup.in.bat`.

The `setup.in` file contains the following variables:

- **`I2A_TOOLKIT_SCRIPTS_JAVA_HOME`**

  The path to a Java installation that is used to run the toolkit tasks.

  The path can be absolute or a relative path from the root of the toolkit.

  By default, it is `\tools\java` on Windows and `tools/java` on Linux.

  Note: When you deploy i2 Analyze, Eclipse Temurin JDK 17 is installed in the default location. If you change the path, you must ensure that an installation of Java 17 exists in the specified location.

- **`I2A_COMPONENTS_JAVA_HOME`**

  An absolute path to a Java installation that is used by the Solr, ZooKeeper, and Liberty components of i2 Analyze. By default, it is `C:\i2\i2analyze\deploy\java` on Windows and `/opt/i2/i2analyze/deploy/java` on Linux.

  Note: When you deploy i2 Analyze, Eclipse Temurin JDK 17 is installed in the specified location if it does not contain an installation of Java.

The following variables are commented out in the file, but can be used to specify different Java installations for the components of i2 Analyze. If the file contains these variables, their values override the value of `I2A_COMPONENTS_JAVA_HOME` for each component.

- **`SOLR_JAVA_HOME`**

  An absolute path to a Java installation that is used by Solr.

- **`ZK_JAVA_HOME`**

  An absolute path to a Java installation that is used by ZooKeeper.

- **`LIBERTY_JAVA_HOME`**

  An absolute path to a Java installation that is used by Liberty.

- **`ETL_TOOLKIT_JAVA_HOME`**

The path to a Java installation that is used by the ingestion commands in the ETL toolkit. The path can be absolute or a relative path from the root of the ETL toolkit.

Note: When you create the ETL toolkit, Eclipse Temurin JDK 17 is included as part of the ETL toolkit. If you change the path, you must ensure that an installation of Java 17 exists in the specified location.

The `setup.in` is part of the configuration, and is copied to each deployment toolkit in your environment. It is recommended that you install Java in the same location on each server where a particular component is located. This means that this file remains the same on each copy of the configuration.

Procedure

1. In a text editor, open the `setup.in` script file for your operating system.
2. Uncomment and update the paths to specify the Java 17 installations that you want to use.
3. Save and close the file.

What to do next

After you update the file, continue with the rest of the deployment process.

Deploying a proxy server for use with i2 Analyze

When you deploy i2 Analyze, the application is bound to the local server on port `9082`. Clients can use the hostname, IP address of the server, or `localhost` to connect to the deployment. To work with your existing network, or for load-balancing purposes, you can deploy a proxy server or load balancer for use with i2 Analyze.

Before you begin

When you deploy a proxy server or load balancer for i2 Analyze, there are two mechanisms that you can use to determine how users connect to the application:

- Populate the `X-Forwarded-Host` and `X-Forwarded-Proto` headers on requests
- Specify a single connection string for the `FrontEndURI` setting in the i2 Analyze configuration

To use a proxy server or load balancer with i2 Analyze, it must meet the following requirements:

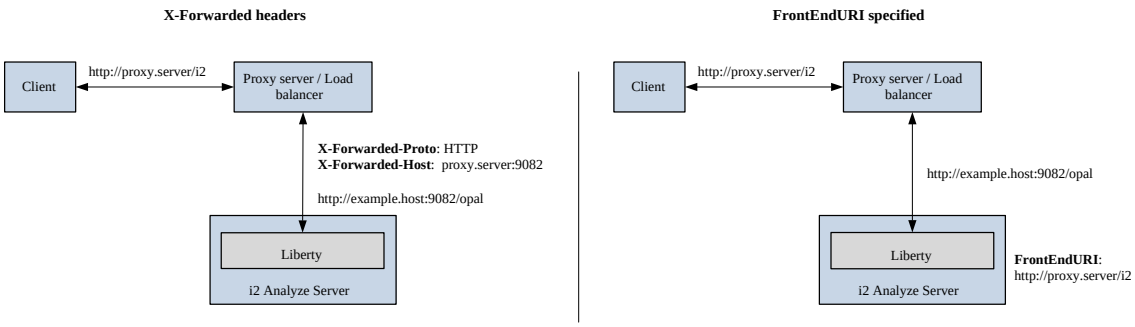
- Handle WebSocket requests and keep the WebSocket connection alive for longer than 180 seconds.
- Pass through any security configurations (for example client certificate authentication) to the i2 Analyze server. You must enable this according to the documentation for your proxy server.
- If you want to use the `X-Forwarded` headers, the proxy server or load balancer must be able to populate them. For more information about the headers, see X-Forwarded-Host and X-Forwarded-Proto.
- If you are planning to deploy i2 Analyze with high availability, your load balancer must also provide server persistence. For more information, see Deploying a load balancer.

About this task

To allow users to connect by using the URI of the proxy server or load balancer, you can populate `X-Forwarded-Host` and `X-Forwarded-Proto` headers or specify the URI in the i2 Analyze configuration. If you configure the connection URI for your deployment, users that connect to i2 Analyze must use the URI that you specify.

If your application has the `http-server-host` attribute set to `true` in the `topology.xml`, your proxy server is not required to route requests to the port number the application is listening on.

The following image shows the URIs that you might use in this example:



Procedure

1. To use the `X-Forwarded` headers to allow clients to connect to the deployment:
   a. Configure your proxy server or load balancer to populate the `X-Forwarded-Host` and `X-Forwarded-Proto` headers with the hostname and protocol that was used to connect to the proxy server or load balancer. For more information about the headers, see X-Forwarded-Host and X-Forwarded-Proto.

   After you configure the X-Forwarded headers, you do not need to redeploy i2 Analyze. The URI that is displayed in the console will remain the same.
2. To specify a single connection URI:
   a. Using a text editor, open the `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoClientSettings.properties` file.
   b. Set the value of the `FrontEndURI` property to the URI that can be used to connect to your deployment of i2 Analyze through the proxy server. For example, `FrontEndURI=http://proxy.server/i2`.
   c. Save and close the file.

   After you update the connection URI, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with your changes. After you deploy i2 Analyze and start the server, the URI that can be used to access the system is displayed in the console.

What to do next

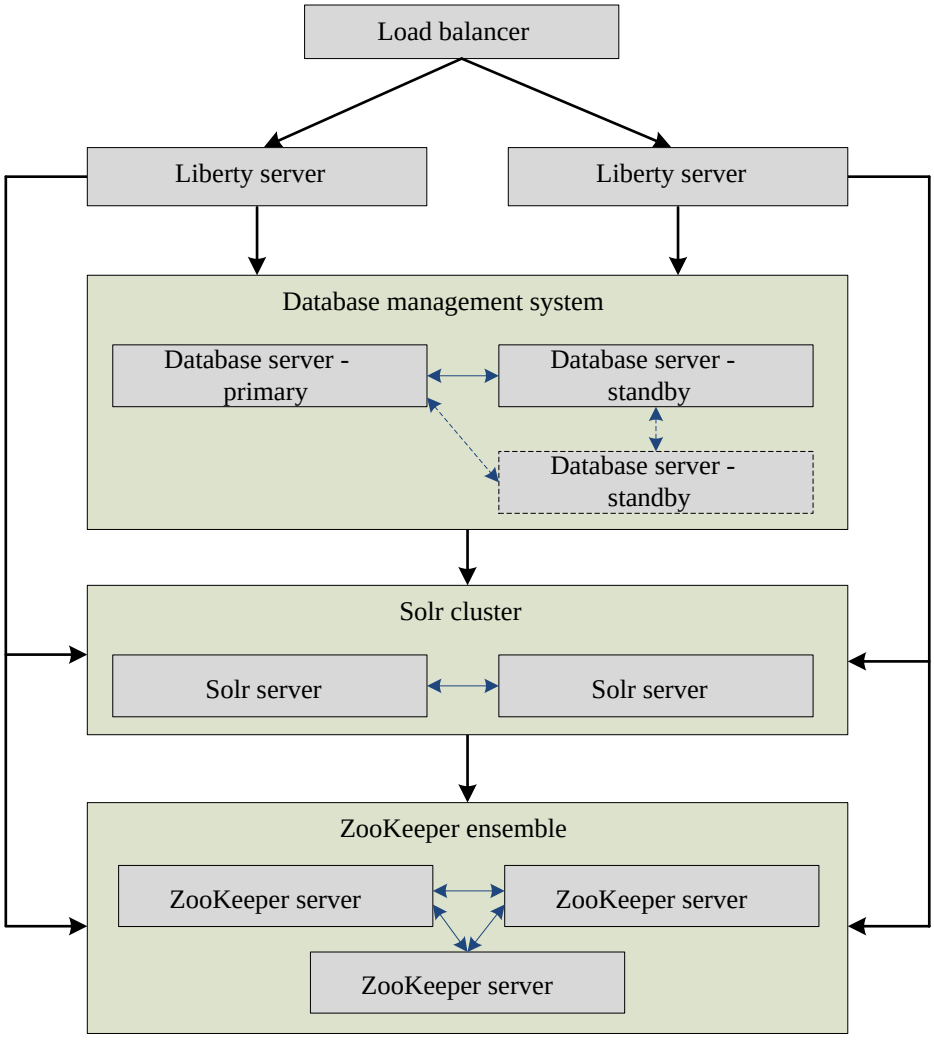
Ensure that you can access i2 Analyze by using this URI from a client workstation that uses the proxy server.

i2 Analyze with high availability

i2 Analyze supports an active/active high availability and disaster recovery deployment pattern. The only component that is not deployed in an active/active state is the database management system, which

uses an active/passive pattern where one instance of the Information Store database is the primary and others are standby.

The following diagram shows the minimum number of servers for a deployment with fault tolerance for one server failure of each component:



After you read the following information that explains how each component is used in a deployment with high availability, follow the instructions to deploy i2 Analyze in this pattern. For more information about how to deploy i2 Analyze, see Deploying i2 Analyze with high availability.

Load balancer

A load balancer is required to route requests from clients to the Liberty servers that host the i2 Analyze application. Additionally, the load balancer is used to monitor the status of the Liberty servers in a deployment. The load balancer must route requests only to servers that report their status as "live".

After you deploy i2 Analyze with a load balancer, you can make requests to i2 Analyze through the load balancer only.

You can also use the load balancer to distribute requests from clients across the servers in your deployment. The load balancer must provide server persistence for users.

Liberty

To provide high availability of Liberty, and the i2 Analyze application, you can deploy i2 Analyze on multiple Liberty servers. Each Liberty server can process requests from clients to connect to i2 Analyze.

In a deployment with multiple Liberty servers, one is elected the *leader*. The leader can process requests that require exclusive access to the database. The actions that are required to be completed on the leader Liberty are described in Liberty leadership configuration.

At a minimum, two Liberty servers are required.

Database management system

To provide high availability of the Information Store database, use the functions provided by your database management system to replicate the primary database to at least one standby instance. i2 Analyze connects to the primary instance at one time, with the contents of the Information Store database replicated to the standby instances.

If the primary instance fails, one of the standby instances becomes the primary. When a standby becomes the primary, it must be configured to be read and writable. This means that i2 Analyze can continue to function when the initial primary server fails.

- For more information about high availability in Db2, see High availability.
- For more information about high availability in SQL Server:
  - For Enterprise edition, see What is an Always On availability group?
  - For Standard edition, see Basic availability groups
  - If you are deploying with SQL Server on Linux, see Availability groups for SQL Server on Linux

For Db2, at a minimum one primary server and one standby server is required.

For SQL Server, at a minimum one primary server and two standby servers are required. If you are using basic availability groups, this is one standby server and one failover quorum witness.

Solr

i2 Analyze uses SolrCloud to deploy Solr with fault tolerance and high availability capabilities. To provide fault tolerance and high availability of the Solr cluster, deploy the cluster across at least two Solr nodes with each node on a separate server.

At a minimum, two Solr servers are required.

For more information about SolrCloud, see How SolrCloud Works.

ZooKeeper

To deploy ZooKeeper for high availability, deploy multiple ZooKeeper servers in a cluster known as an *ensemble*. For a highly available solution, Apache ZooKeeper recommend that you use an odd number of ZooKeeper servers in your ensemble. The ZooKeeper ensemble continues to function while there are more than 50% of the members online. This means, that if you have three ZooKeeper servers, you can continue operations when a single ZooKeeper server fails.

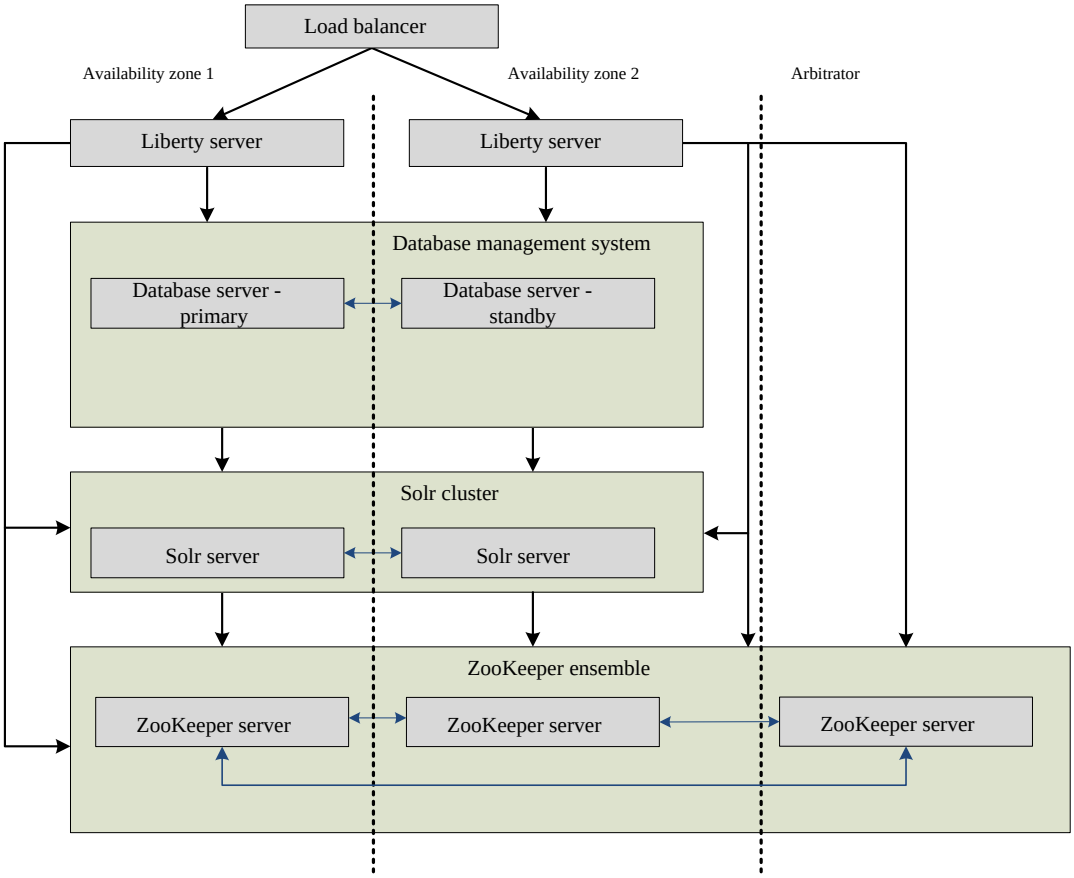
At a minimum, three ZooKeeper servers are required.

For more information about a multiple server ZooKeeper setup, see Clustered (Multi-Server) Setup.

Deploy for disaster recovery

The deployment topology to provide availability in the event of an availability zone failure is similar to the high availability topology. An active/active pattern is still used for Liberty, Solr, and ZooKeeper, however the servers that you deploy are in separate availability zones.

The components behave in the same way as in the high availability deployment topology, except they are deployed in availability zones. You can think of an availability zone as a data center. By deploying across a number of availability zones, the deployment continues to function if one of the zones fails.



Note: For Microsoft SQL Server, you must have a third database server to act as a witness. To continue to provide availability when one of the zones fails, this server must be located outside of the two availability zones. For more information, see Three synchronous replicas.

- Because ZooKeeper requires more than 50% of the servers in an ensemble to be available, one server must be located outside of the availability zones. The ZooKeeper server does not require a significant amount of resources so it can be located on any server that can communicate with the availability zones.
- You can deploy multiple Liberty and Solr servers to provide more availability at a server level within each zone.

- For the components that use an active/active pattern, servers in each availability zone are used to process requests. To maintain the performance of the system, ensure that the network that is used between availability zones is stable and provides enough bandwidth for system operation.

Deploying i2 Analyze with high availability

When you deploy i2 Analyze with high availability, there are a number of steps that you must complete.

About this task

To configure i2 Analyze for high availability, there are a number of extra configuration and deployment steps that you must complete in addition to a normal multiple server deployment. When you are deploying i2 Analyze into production with a high availability configuration, complete the following steps when you are in your pre-production, test, and production environments. After you create the high availability configuration, you can copy it between environments as usual.

Procedure

1. Install any prerequisite software to prepare your servers for the pre-production environment.
   - For a deployment with high availability, use the multiple servers deployment topology with at least the minimum number of servers required for high availability of each component. For more information, see Deployment topologies.
   - To deploy with high availability, you must configure your database management system after you install it:
     - Preparing Db2 for HADR
     - Preparing SQL Server for HADR
2. Copy the `toolkit\configuration` directory from the configuration development environment, to the `toolkit` directory at the root of the deployment toolkit on one of the Liberty servers.
3. Update the configuration to specify your remote database, follow the instructions in Specifying remote database storage to update the `environment.properties` and `topology.xml` files.
   a. If you are using Db2, you can specify any alternative database location information in the `topology.xml`.
      - If you are using an automated cluster-controller such as IBM Tivoli System Automation for Multiplatforms (TSAMP), then specify the hostname and port number of the cluster.
      - If you are not using an automated cluster-controller, in the `<database>` element for your Information Store, provide the hostname and port number of any standby databases in the `alternate-hosts` attribute.

        The host and port number must be separated by a colon (`:`), and each database server must be separated by a (`,`).

        For example, `alternate-hosts="hostname:port_number,hostname:port_number"`.
   b. If you are using SQL Server, specify the hostname and port number of the availability group listener.
4. Update the configuration to configure each component for high availability:
   a. Configuring ZooKeeper for HADR
   b. Configuring Solr for HADR
   c. Deploying a load balancer

The `environment.properties`, `http-server.properties`, and `topology.xml` contain hostname and file path settings that you might need to update for the servers in your pre-production environment. For more information, see Configuration files reference.

5. Deploy and start i2 Analyze with one Liberty server by following the instructions in Deploying i2 Analyze on multiple servers.
6. After you deploy with one Liberty server, update the configuration on any other Liberty servers.
   a. Copy the `configuration` directory from the deployed Liberty server to the `toolkit` directory on any other Liberty servers in your environment.
   b. On each Liberty server, in the `topology.xml` file update the `host-name` and `port-number` attributes of the `<application>` element.
7. Deploy and start the i2 Analyze application on each of the other Liberty servers:

```
setup -t installLiberty
setup -t deployLiberty
setup -t startLiberty
```

After you deploy i2 Analyze, you can replicate any configuration changes that are not stored in the configuration of i2 Analyze on each Liberty server.

The security configuration must be the same on each server.

1. Configure Liberty security for your environment. To do this, repeat any changes that you made to the Liberty configuration in the previous environment. This might involve copying the user registry file, or updating the `server.xml` file.
2. After you deploy i2 Analyze, configure your database management system to replicate the Information Store database to your standby servers.
   a. If you are using Db2, configure high availability for the Information Store database and replicate it to any standby database instances. For more information, see Replicate the Information Store in Db2.
   b. If you are using SQL Server, add the Information Store to your availability group. For more information, see Replicate the Information Store in SQL Server.
3. Complete any configuration changes in the Information Store database on the primary server.
   a. If you created any rules or schedules to delete records by rule, replicate the rules and schedules that you created in the previous environment. On SQL Server, you must update the automated job creation schedule on every database instance. For more information see, Changing the automated job creation schedule.
   b. If you created any merged property values definition views for your ingestion process, replicate the view definition that you created in the previous environment.

What to do next

After you deploy i2 Analyze with high availability, return to perform the rest of the instructions for creating a deployment in your current environment:

- Creating the pre-production environment
- Creating the test environment
- Creating the production environment

Preparing Db2 for HADR

When you deploy i2 Analyze with multiple database servers, there must be a mechanism in place to route requests to the new primary server if the current primary server fails.

About this task

In Db2, you configure the Information Store database for HADR after you create it. However, there are some decisions that you must make before you create the i2 Analyze configuration.

When you configure i2 Analyze, you must specify the location of the remote database servers in your system. When you configure Db2 for HADR, you can use an automated cluster-controller such as IBM Tivoli System Automation for Multiplatforms (TSAMP) to ensure that clients can connect to the new primary database instance.

If you do not choose to use an automated cluster-controller, you can specify the connection properties for all of your database servers in the i2 Analyze configuration. When you provide the details in the configuration, i2 Analyze uses client-side Automatic Client Rerouting (ACR) to route requests to the new primary server.

Before you start to configure i2 Analyze, choose an approach to ensure that clients can connect to any standby database instances.

- For more information about automatic cluster-controllers, see High availability through clustering.

What to do next

Deploy i2 Analyze and the Information Store database. For more information, see Deploying i2 Analyze with high availability.

Preparing SQL Server for HADR

To deploy i2 Analyze with HADR for the Information Store database, SQL Server must be configured to be part of an availability group.

Before you begin

Ensure that SQL Server is installed correctly for Always On availability groups. For more information, see Installing Microsoft SQL Server for i2 Analyze.

If you are using Microsoft SQL Server Standard edition, you can use basic availability groups only. For more information about basic availability groups, see Basic Always On availability groups.

To configure SQL Server for high availability, you add your database servers to a fail over cluster and then add any database instances to an Always On availability group.

Deploying SQL Server on Windows

On Windows, use Windows Server failover clustering to create a cluster of servers to host your SQL Server instances. For more information, see Failover Clustering in Windows Server.

1. Install the Windows Failover Clustering feature on each of the servers that will be in your SQL Server Always On Availability Group and create a failover cluster.

   For more information, see Create a failover cluster.

   For i2 Analyze, the failover cluster must use a static IP address.
2. Create and configure an Always On availability group.

For more information, see Reference for the creation and configuration of Always On availability groups

a. Enable Always On availability groups on each database server.

For more information, see Enable the Always On availability group feature for a SQL Server instance.

b. Create an Availability Group on the primary SQL Server instance.

For more information, see Use the New Availability Group Dialog Box (SQL Server Management Studio).

- Set **Required synchronized secondaries to commit** to 1.

  Set it to 0 if you are using Basic availability groups.
- Select **Database level health detection**.
- Do not specify any **Availability Databases** now because the Information Store database does not exist.

c. Create an availability group listener for the availability group.

For more information, see Configure a listener for an Always On availability group.

Use a static IP address for the availability group listener. When you configure i2 Analyze to connect to a remote Information Store database, you specify the hostname and port number of the Availability Group Listener.

Deploying SQL Server on Linux

On Linux, use Pacemaker to create the server failover cluster.

1. Deploy a Pacemaker cluster.

   For more information, see Deploy a Pacemaker cluster for SQL Server on Linux.
2. Create and configure an Always on availability group and the Pacemaker cluster.

   For more information, see Create and configure an availability group for SQL Server on Linux.

   - Set cluster type to EXTERNAL
   - In Select Databases, do not select any databases because the Information Store database does not exist.
   - In Specify Replicas, create an Availability Group Listener. When you create the availability group listener, use a static IP address. When you configure i2 Analyze to connect to a remote Information Store database, you specify the hostname and port number of the Availability Group Listener.
   - When you create the IP address resource for the availability group, use the same IP address as the Availability Group Listener.

Deploy i2 Analyze and the Information Store database. For more information, see Deploying i2 Analyze with high availability.

Configuring ZooKeeper for HADR

The i2 Analyze configuration specifies the structure of your ZooKeeper ensemble. To provide high availability, configure a ZooKeeper ensemble with an odd number of severs.

About this task

In the i2 Analyze configuration, provide the details about the ZooKeeper servers in your environment.

Procedure

Edit the `configuration\environment\topology.xml` file to specify your ZooKeeper servers.

For example:

```
<zookeeper id="zoo">
  <zkhosts>
    <zkhost
      host-name="<zookeeper_server1_host_name>" id="1"
      port-number="9983" quorum-port-number="10483" leader-port-
number="10983"
      data-dir="C:/i2/i2analyze/data/zookeeper"
      />
    <zkhost
      host-name="<zookeeper_server2_host_name>" id="2"
      port-number="9983" quorum-port-number="10483" leader-port-
number="10983"
      data-dir="C:/i2/i2analyze/data/zookeeper"
      />
    <zkhost
      host-name="<zookeeper_server3_host_name>" id="3"
      port-number="9983" quorum-port-number="10483" leader-port-
number="10983"
      data-dir="C:/i2/i2analyze/data/zookeeper"
      />
  </zkhosts>
</zookeeper>
```

Where `<zookeeper_serverx_host_name>` is the hostname of the ZooKeeper server.

Each `<zkhost>` element must have a unique value for the `id` attribute.

For more information about the possible values for each attribute, see Solr and ZooKeeper.

What to do next

Continue configuring the i2 Analyze configuration. For more information, see Deploying i2 Analyze with high availability.

Configuring Solr for HADR

The i2 Analyze configuration specifies the structure of your Solr cluster. To provide high availability, configure a Solr cluster over at least two Solr servers.

About this task

In the i2 Analyze configuration, provide the details about the Solr servers in your environment. To provide high availability of the content of the Solr indexes, specify the number of replicas, the minimum replication factor, and the location of the replicas.

For more information about SolrCloud, see How SolrCloud Works.

Procedure

1. Specifying the Solr nodes.
   To deploy Solr for high availability, you must have at least two Solr nodes on separate hosts.

   a. Add the Solr nodes to your `topology.xml` file.
      For example:

```
<solr-nodes>
  <solr-node
    memory="2g"
    data-dir="C:/i2/i2analyze/data/solr"
    host-name="solr_server1_host_name"
    id="node1"
    port-number="8983"
  />
  <solr-node
    memory="2g"
    data-dir="C:/i2/i2analyze/data/solr"
    host-name="solr_server2_host_name"
    id="node2"
    port-number="8983"
  />
</solr-nodes>
```

      Where `solr_serverx_host_name` is the hostname of a Solr server.
      For more information about the possible values for each attribute, see Solr and ZooKeeper.

2. Configuring Solr replicas.
   In Solr, the data is stored as `documents` in `shards`. Every shard consists of at least one `replica`. For a highly available solution, you must have more than one replica of each shard and these replicas must be distributed across the servers that host the Solr nodes.

   a. Specifying the `replication factor` in your `topology.xml`.
      The replication factor is the number of replicas to be created for each shard. For high availability, this must be 2 or more. You specify the replication factor in the `num-replicas` attribute of the `<solr-collection>` element.

   b. Specifying the `minimum replication factor` in your `topology.xml`.
      The minimum replication factor defines when data is successfully replicated in Solr. If you have three replicas for a shard and a minimum replication factor of 2, a write operation is deemed successful if the data is written to at least two replicas. You specify the minimum replication factor in the `min-replication-factor` attribute of the `<solr-collection>` element.
      The following extract from a `topology.xml` file shows an example of the `num-replicas` and `min-replication-factor` attributes:

```
<solr-collections>
  <solr-collection
  num-replicas="2"
  min-replication-factor="1"
  id="main_index"
  type="main"
  num-shards="1"
  />
...
</solr-collections>
```

3. To configure the replica placement plugins, modify the `configuration/solr/solr.replica.placement.plugin.json` file.

   By default, i2 Analyze uses Solr's Affinity Placement plugin to place the Solr replicas. This plugin attempts to avoid placing replicas on a single node. To aid the placement of replicas across your Solr servers, the host name of your Solr nodes is used as an availability zone.
   For more information about the Solr plugins, see Replica Placement Plugins.

   The replica placement is updated when you create the Solr cluster as part of the deployment steps. Additionally, you can update the replica placement by running the `updateSolrReplicaPlacementPlugin` toolkit task.

What to do next?

Continue configuring the i2 Analyze configuration. For more information, see Deploying i2 Analyze with high availability.

Deploying a load balancer

In a deployment with HADR, there are multiple Liberty servers that host the i2 Analyze application. To enable analysts to connect to the deployment by using a single URI, you must use a load balancer to route requests to each Liberty server in your deployment.

About this task

For a load balancer to work successfully with i2 Analyze, it must be configured to:

- Handle WebSocket requests.
- Route requests to the i2 Analyze application on active Liberty servers only.
- Use a cookie issued by the load balancer to achieve server persistence for a user. The server persistence should last as long as possible or until the server is not active.

If the connection from the clients is secured by using SSL, you might need to implement SSL termination in the load balancer to achieve server persistence.

You can also configure your load balancer to balance requests across the Liberty servers in your deployment. The mechanism that you choose must allow for server persistence to be maintained.

For more information about load balancers and application layer server persistence, see Load Balancing, Affinity, Persistence, Sticky Sessions: What You Need to Know.

Procedure

1. Install your load balancer on a separate server from any other component in the deployment.
2. Configure your load balancer to identify the active Liberty servers by using the `api/v1/health/live` REST endpoint.

   The `live` endpoint returns `200` when the server is live and processing requests. For more information about the endpoint, see The `health/live` endpoint.

   Call the `live` endpoint on each Liberty server in your deployment. The endpoint returns the status of a single Liberty server, and not the others in the deployment.
3. Configure your load balancer to use cookie-based server persistence.

When a user connects, if there is no cookie from the load balancer that indicates it has already connected to one of the active Liberty servers, provide it with one. When this user connects again, the cookie it received ensures that any requests are routed to the same Liberty server.

If the server that a user has persistence with is offline, the load balancer must route the request to a live Liberty server. The user's cookie must be updated to achieve persistence with the live server that it connected to.

4. Before you can connect to i2 Analyze via the load balancer, you might need to specify the connection URI that clients can use to connect to i2 Analyze.

   a. In the `configuration\fragments\opal-services\WEB-INF\classes\DiscoClientSettings.properties` file, set the value of the `FrontEndURI` setting to the URI that can be used to connect to your deployment of i2 Analyze.

For more information, see Specifying the connection URI. Some load balancers modify the HTTP origin header, the value that you specify for the `FrontEndURI` must match the value of the HTTP origin header after it is modified by the load balancer.

What to do next

Continue configuring the i2 Analyze configuration. For more information, see Deploying i2 Analyze with high availability.

Replicate the Information Store in Db2

After you create the Information Store database on the primary database server, initialize HADR and replicate the Information Store database to your standby servers.

Procedure

To initialize HADR for the Information Store, follow the instructions described in Initializing high availability disaster recovery (HADR).

When you configure the synchronize mode, it is recommended that you use the SYNC synchronize mode. For more information about the synchronization modes, see High availability disaster recovery (HADR) synchronization mode.

At a high level, the steps are:

- Update the Information Store database configuration for HADR
- Create a backup of the Information Store database on the primary server
- Restore the backup on the standby servers
- Update the Information Store database configuration on the standby servers
- Start HADR as standby on the standby servers
- Start HADR as primary on the primary server

What to do next

Continue the deployment process. For more information, see Deploying i2 Analyze with high availability.

Replicate the Information Store in SQL Server

After you create the Information Store database on the primary database server, add the database to your availability group and replicate the Information Store database to your standby servers.

Procedure

1. Before you can add the Information Store database to your availability group, you must create a full backup. For more information, see Create a Full Database Backup.
2. Add the Information Store database to your availability group. For more information, see Add a Database to an Always On availability group.
3. Add any replicas to the availability group. For more information, see Add a secondary replica to an Always On Availability Group. When you add the replicas to the availability group, configure the replicas as follows:
   - Set **Failover Mode** to **Automatic**
   - Set **Availability Mode** to **Synchronous commit**
   - Set **Readable Secondary** to **Yes**

     If you are using Basic availability groups, set it to **No**.

What to do next

Continue the deployment process. For more information, see Deploying i2 Analyze with high availability.

i2 Analyze deployment commands

The scripts that deploy i2 Analyze depend on the values in the i2 Analyze configuration. The scripts and commands that you run depend on the topology of the deployment that you want to create.

Deploying i2 Analyze

To deploy i2 Analyze in a single-server deployment topology, you can run a script. After i2 Analyze is successfully deployed, you can start the system.

Before you begin

You must have an i2 Analyze configuration that is set up for a single-server, remote database only, or i2 Connect gateway only deployment topology. For more information about creating a valid configuration, see Creating the pre-production environment.

Run any toolkit commands from the `toolkit\scripts` directory in the deployment toolkit.

Procedure

1. Deploy i2 Analyze:

   ```
   setup -t deploy
   ```
2. Start i2 Analyze:

   ```
   setup -t start
   ```

What to do next

If an error message is displayed, refer to Troubleshooting the deployment process.

After you deploy and start i2 Analyze, return to perform the rest of the instructions for creating a deployment in your current environment:

- Creating the pre-production environment
- Creating the test environment
- Creating the production environment
- Deploying i2 Analyze with high availability

Deploying i2 Analyze on multiple servers

To deploy i2 Analyze in a multiple-server deployment topology, you must run the commands to install, deploy, and start the components of i2 Analyze on each server.

Before you begin

You must have an i2 Analyze configuration that is set up for a multiple-server physical deployment topology or a deployment with high availability:

- For more information about creating a valid multiple-server configuration, see Creating the pre-production environment.
- For more information about creating a valid high availability configuration, see Deploying i2 Analyze with high availability.

About this task

To deploy i2 Analyze in a multiple server deployment topology, you must provide the configuration to each deployment toolkit. Then, you can run the commands to deploy the components of i2 Analyze on each server. It is important to pay attention to which server you must run each command on, and whether you need to specify the hostname of that server.

Run any toolkit commands from the `toolkit\scripts` directory in the deployment toolkit on the specified server in your environment.

Copying the i2 Analyze configuration

The i2 Analyze configuration is required by all servers that host components of i2 Analyze. You do not have to copy the configuration to the database server, if it contains only the Information Store database and no other components.

Procedure

Copy the `toolkit\configuration` from the server where you created and populated your i2 Analyze configuration, to the `toolkit` directory of the deployment toolkit on each server in your environment.

Installing components

Install the components of i2 Analyze on the servers that you have identified.

Procedure

1. On the Liberty server, run the following commands:

   ```
   setup -t installLiberty
   ```

2. On each ZooKeeper server, run the following command:

   ```
   setup -t installZookeeper
   ```

3. On each Solr server, run the following command:

   ```
   setup -t installSolr
   ```

Deploying and starting components

Deploy and start the components of i2 Analyze on the specified servers.

Procedure

1. On each ZooKeeper server, encode the credentials and create and start any ZooKeeper hosts:

   ```
   setup -t ensureCredentialsEncoded
   setup -t createZkHosts --hostname <zookeeper.hostname>
   setup -t startZkHosts --hostname <zookeeper.hostname>
   ```

   Where `zookeeper.hostname` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the `topology.xml` file.

2. On the Liberty server, run the command to upload the Solr configuration to ZooKeeper:

   ```
   setup -t createAndUploadSolrConfig --hostname <liberty.hostname>
   ```

   Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

3. On each Solr server, encode the credentials and create and start any Solr nodes:

   ```
   setup -t ensureCredentialsEncoded
   setup -t createSolrNodes --hostname <solr.hostname>
   setup -t startSolrNodes --hostname <solr.hostname>
   ```

   Where `solr.hostname` is the hostname of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

On the Liberty server, run the commands to deploy and start a number of the components.

1. Create the Solr collections:

   ```
   setup -t createSolrCollections --hostname <liberty.hostname>
   ```

   To test that the Solr Collection is created correctly, click **Cloud** in the Solr Web UI, or you can go to `http://solr.hostname:port-number/solr/#/~cloud`. Log in with the user name and password for Solr in the `credentials.properties` file.

   A horizontal tree with the collection as the root is displayed. Here you can see the breakdown of the shards, nodes, and replicas in any collections.

2. Create the Information Store database:

   ```
   setup -t createDatabases
   ```

   To check that the database is created correctly, connect to the database by using a database management tool.

3. Deploy the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```

4. If you are using IBM HTTP Server, configure the HTTP Server:

   ```
   setup -t configureHttpServer
   ```

5. Start the Liberty server:

```
setup -t startLiberty
```

What to do next

After you deploy and start i2 Analyze, return to perform the rest of the instructions for creating a deployment in your current environment:

- Creating the pre-production environment
- Creating the test environment
- Creating the production environment
- Deploying i2 Analyze with high availability

Troubleshooting the deployment process

i2 Analyze provides mechanisms that assist with deployment before, while, and after you do it.

Deployment toolkit validation messages

The i2 Analyze deployment toolkit validates some of the configuration files that are required for a deployment. By reading the validation messages, you can be certain that required properties have values, that property values have the correct format, and that required files are in the correct locations.

Validation takes place when you run a number of the deployment toolkit tasks. Validation can complete in the following ways:

Success

On success, the validation process displays a brief configuration summary, and the main deployment process continues. For example:

```
============== Configuration Summary (brief) ==============
 + configuration/fragments/common/WEB-INF/classes/ApolloServerSettings
Mandatory.properties:
   - ChartingSchemesResource='law-enforcement-schema-charting-schemes.xml'
   - SchemaResource='law-enforcement-schema.xml'
   - DynamicSecuritySchemaResource='security-schema.xml'
============================================================
```

Warning

When there is a warning, the validation process displays a brief configuration summary and a `WARNINGS` section. The `WARNINGS` section identifies settings that might not be configured correctly, but the deployment process continues. For example:

```
====================== WARNINGS (1) ======================
 + configuration/fragments/common/WEB-INF/classes/ApolloServerSettings
Mandatory.properties:
   - The <SchemaResource> property has not been set.
============================================================
```

Here, the schema is not set, and so the default law enforcement schema is used.

Error

If an error occurs, the validation process displays a longer configuration summary, and an `ERRORS` section. The `ERRORS` section identifies missing values that must be present. The deployment process stops, and you must correct the errors before you attempt to deploy again. For example:

```
====================== ERRORS (1) ======================
 + configuration/environment/opal-server/environment.properties:
```

```
   - "db.database.location.dir" has not been set
============================================================
```

Here, the database location directory is not set, so the database cannot be configured.

Deployment progress messages

During the deployment process, i2 Analyze displays detailed messages that provide information about the state and configuration of the system.

The output from the deployment process describes each task that the `setup` command performs during deployment. If a task runs successfully, then only its name appears. For example:

```
:buildApplication
```

There are two reasons why a task might not run, but deployment can still proceed.

- **`UP-TO-DATE`**

  The task was performed earlier, or its output is already present. For example:

  ```
  :installJDBCDrivers UP-TO-DATE
  ```

- **`SKIPPED`**

  The task is not required for this deployment. For example:

  ```
  :importLTPAKey SKIPPED
  ```

If an error occurs, deployment stops in a controlled manner. i2 Analyze displays a stack trace that contains the name of the task that failed, and information about the location of the error. For example:

```
:createDatabasesIfNecessary FAILED

FAILURE: Build failed with an exception.

* Where:
Script 'C:\i2\i2analyze\toolkit\scripts\gradle\database.gradle' line: 173

* What went wrong:
Execution failed for task ':createDatabasesIfNecessary'.
```

The messages are displayed on screen and sent to the log files that are stored in the `toolkit\configuration\logs` directory.

Deployment log files

i2 Analyze produces logging information about deployment tasks and the transactions that take place when the system is operational. After deployment, you can check these logs to help diagnose potential issues.

The following types of logged information are available for you to review:

Deployment logs

Each time that you run the `setup` command, a log file is created in the deployment toolkit. The messages in these logs describe which tasks were called, whether the tasks completed successfully, and the details of any issues that occurred.

These log files contain the same information as the console output. You can find them in the `toolkit\configuration\logs` directory.

Liberty logs

In addition to the information in the deployment logs, extra information that relates to the application server is also logged in `deploy\wlp\usr\servers\opal-server\logs`.

The directory contains the following log files that you might use to help troubleshoot any issues:

- **`console.log`**

  The `console.log` file contains basic server status and operation messages, which are also displayed on the console.

- **`\messages\messages.log`**

  The `messages.log` file contains messages from Liberty.

- **`\messages\ffdc\ffdc.log`**

  The `ffdc.log` file contains Fist Failure Data Capture (FFDC) output from Liberty components (for example the database drivers that typically include selective memory dumps of diagnostic data).

- **`\opal-services\i2_Alert_Scheduler.log`**

  The `i2_Alert_Scheduler.log` contains messages that record when the scheduled operations for saved Visual Queries run.

- **`\opal-services\i2_Analysis_Repository.log`**

  The `i2_Analysis_Repository.log` file contains messages that are intercepted by log4j from the i2 Analyze application.

- **`\opal-services\i2_General.log`**

  The `i2_General.log` file contains any messages that are intercepted by log4j, but not from the i2 Analyze application.

- **`\opal-services\i2_Component_Availability.log`**

  The `i2_Solr_Cluster_Status.log` file contains entries for changes to the state of the Solr clusters in your deployment.

- **`\opal-services\i2_Connect_Async_Query.log`**

  The `i2_Connect_Async_Query.log` file contains messages for failures or issues related to asynchronous queries from i2 Connect connectors.

- **`\opal-services\i2_Update_Live_Configuration.log`**

  The `i2_Update_Live_Configuration.log` file contains messages that are displayed in the console when you run the `updateLiveConfiguration` toolkit task or use the `admin/reload` endpoint.

- **`\opal-services\i2_Solr.log`**

  The `i2_Solr.log` file contains messages that are directly from Solr and ZooKeeper.

Solr logs

By default, information that relates to Solr is logged in `deploy\solr\server\logs\<node port>`.

Resolving deployment errors

The sections in this topic describe how to react to some common error and warning messages that can appear during i2 Analyze deployment.

"The results configuration file contains an invalid property type identifier"

The following message is displayed if there are any identifiers in your results configuration file that are not present in the i2 Analyze schema when you start i2 Analyze.

```
# [opal-services-is] OneOffStartupServiceInitializer encountered a problem
 communicating with the database that it cannot recover from.
# [opal-services-is] Exception during initialization. The application is in
 an unusable state.
# [opal-services-is] Errors occurred accessing system resources from
 ApplicationLifecycleManager:
# [opal-services-is] The results configuration file contains an invalid
 property type identifier: ADD1.
```

To resolve this issue, ensure that all of the identifiers in your results configuration file are present in the i2 Analyze schema. For more information, see Setting up search results filtering.

"Default security dimension values provider specifies dimension values that are not in the security schema" or **"Default security dimension values provider specified no values for the dimensions with these IDs"**

These messages can appear when the security dimension values that are specified in the security schema for the `<DefaultSecurityDimensionValues>` element are incorrect. For more information, see Setting default dimension values.

"Unable to determine the DB2 version as you do not have execute permission to db2level"

This message is displayed if the JDBC driver is not present, or the installation path to Db2 is specified incorrectly when you run the `generateDefaults` toolkit task.

There are two ways to resolve this issue:

- Ensure that you provide the JDBC driver for your deployment. For more information, see Specifying the JDBC driver.
- Ensure that the value for the `db.installation.dir` setting in the `environment.properties` file is correct. For more information, see `environment.properties`.

"Invalid attribute value for 'name' in element 'complexType'. Recorded reason: cvc-datatype-valid.1.2.1: 'Direcci##nItemClass' is not a valid value for 'NCName'."

This message is displayed if you try to deploy with a highlight query configuration that contains an invalid character for the Db2 code page. After you install Db2, you specify the code page that the Db2 instance uses. For more information about the value that you must use for the code page, see the Post-install section of Installing IBM Db2 for i2 Analyze.

"Some item types have multiple property types with the same semantic type ID, which is not permitted."

This message appears if the server detects that the Information Store schema assigns the same semantic property type to more than one property type of the same item type. The message includes the types involved and the identifier of the semantic type. For example:

```
Property (ET9)
        Property Type (PRO2): guidc329f90d-3bf7-40c3-8d87-f6fe3cb8d5d4
        Document Type (PRO30): guidc329f90d-3bf7-40c3-8d87-f6fe3cb8d5d4
```

To resolve the problem, open the schema in Schema Designer and make one of the following changes:

- Remove the semantic property type assignment from all but one of the indicated property types
- Assign different semantic property types from the i2 Semantic Type Library to all but one of the indicated property types
- Derive appropriate custom semantic property types from the duplicated type, and assign the derived types to the property types instead.

If the property types in question *are* semantically related, the second and third options provide the best analytical experience.

"<Permission> elements in the security schema have Level attributes ('NONE') that are not valid. Ensure that all permission levels are valid."

This message appears if the server detects that the security schema includes the NONE access level in a permission assignment. For example:

```
<SecurityPermissions>
  <GroupPermissions UserGroup="Analyst">
    <Permissions Dimension="SD-SC">
      <Permission DimensionValue="HI" Level="UPDATE"/>
      <Permission DimensionValue="OSI" Level="NONE"/>
    </Permissions>
  </GroupPermissions>
  ...
```

To resolve the problem, remove the NONE assignment , or replace it with a valid level (UPDATE or READ_ONLY).

Connecting clients

After you deploy and start i2 Analyze, connect to your system by using one of the supported clients and search for data. Depending on the deployed components, you can access an i2 Analyze system by using either i2 Analyst's Notebook or one of the web clients.

Before you begin

Before you can use any client to connect, you must:

- Set up at least one user on the application server that has permission to access records in i2 Analyze
- Ensure that i2 Analyze is started.
- Make a note of the URI for connections. When you start i2 Analyze, the URI that can be used to connect is displayed in the console. For example:

  ```
  Web application available (default_host): http://host_name:9082/opal
  ```

To use Analyst's Notebook to connect, you must also:

- Install Analyst's Notebook. For more information, see the i2 Analyst's Notebook documenation about installing Analyst's Notebook and working with i2 Analyze.

To use a web client to connect, you must also:

- Ensure that you have the correct license agreements in place to use the i2 Investigate or the i2 Notebook web client.

- If you want to use the i2 Notebook web client, ensure that you have configured your deployment to use SSL and given users the relevant command access permission. For more information, see Enabling access to the i2 Notebook web client.

  Note: When you create an example deployment, the example user already has the necessary permission to use the i2 Notebook web client. Also, if you have access to the i2 Analyze server, you can connect to the client through the `localhost` URL without configuring SSL.

Procedure

To use a web client to connect:

1. Open a web browser, and navigate to `https://<host_name>/opal`. The web client displays a login dialog.
2. Enter the name and password of a user who is registered in the application server.

   If that user has permission to use the i2 Notebook web client, you see that application's user interface. If they do not have permission, you see the i2 Investigate web client user interface instead.
3. Search for and visualize data to verify that the application is working correctly.

What to do next

i2 Analyze can display a welcome page when users connect through Analyst's Notebook. For information on setting that up, see Displaying a welcome page in Analyst's Notebook.

Removing a deployment of i2 Analyze

You might need to reset a deployment of i2 Analyze to its just-installed state. For example, you might complete an example deployment and then want to start creating a production deployment on the same system.

Before you begin

Important: This procedure completely removes all the data and deployed components of i2 Analyze. If you want to retain any data from the system, back up your data first. For more information about backing up your data, see Backing up a deployment.

About this task

To remove a deployment of i2 Analyze, you must remove the database from the system and then the deployment and data files from disk.

Procedure

1. Stop the deployment of i2 Analyze:
   - In a single-server or remote database topology, run `setup -t stop`.
   - In a multiple-server topology, see Stopping and starting i2 Analyze.
2. If your deployment contains a data store, drop the database from the system:

   ```
   setup -t dropDatabases
   ```

   A message is displayed when you run the toolkit task to confirm that you want to complete the action. Enter Y to continue.

To ensure that the database is removed correctly, use a database management tool to try and connect to the database.

3. Move or delete the deployment and data directories in every i2 Analyze deployment toolkit in your environment. By default, the directories are: `i2\i2analyze\deploy` and `i2\i2analyze\data`:
   - The deployment directories for the components of i2 Analyze, and the data directory for i2 Analyze is specified in the `environment.properties` file. For more information, see `environment.properties`.
   - The data directories for the components of i2 Analyze are specified in the `topology.xml` file. For more information about this file, see `topology.xml`.
4. Move or delete the `toolkit\configuration` directory in every i2 Analyze deployment toolkit in your environment.

Results

The i2 Analyze deployment is returned to its just-installed state.

Best practices

When you create or manage a production deployment of i2 Analyze, there are a number of best practices that you might follow.

Server host names

When you have multiple environments with i2 Analyze deployed, it can be useful to use host names for the servers and application that identify their purpose. For example, in your development environment you might set the host names of servers as follows: `i2analyze.development.server`.

Configuration management

When you create and maintain a deployment of i2 Analyze, it is an iterative process. Because of the iterative nature, it is important to keep a record of changes to the i2 Analyze configuration. By using a source control system, you can maintain a history of changes to the configuration.

After you populate the configuration directory in the pre-production environment, make a copy of the directory in a location that is managed by a source control system.

Permanent environments

You can maintain your final development and test environments so that you can return to previous deployment phases or use the environments to test any future upgrades of the system.

Production changes

Make any configuration changes in the lowest available deployment, then promote up to production.

Installing i2 Analyze

i2 distributes i2 Analyze in an archive file that contains the server software and tested versions of its prerequisites. For some deployments, installing i2 Analyze just involves extracting the files from the archive. For others, you need to add a supported database.

Before you begin

You must ensure that the system that you are planning to install i2 Analyze onto is compatible with the system requirements. For details of the system requirements, see the release notes that are available from the i2 Analyze support page.

To install i2 Analyze, you must have the i2 Analyze version 4.4.4 distribution. Choose one of the following distributions to install i2 Analyze from:

- i2 Analyze V4.4.4 (Archive install) for Windows
- i2 Analyze V4.4.4 (Archive install) for Linux

About this task

i2 Analyze is provided in a `.zip` archive file for Windows, and a `.tar.gz` archive file for Linux. To install i2 Analyze, you need to extract the archive file and then accept the license agreement.

Procedure

1. Download the i2 Analyze distribution file for your operating system, and extract the contents into one of the following directories:
   - On Windows, `C:\i2\i2analyze`
   - On Linux, `/opt/i2/i2analyze`

   Before you can use i2 Analyze, you must read and accept the license agreement and copyright notices.
2. In a text editor, open the `notices` file and the license file for your language from the `i2analyze\license` directory.

   For example, the English license is in the `LA_en` file.
3. Accept the license and copyright notices.
   a. Open the `i2\i2analyze\license_acknowledgment.txt` file.
   b. To accept the license and copyright notices, change the value of `LIC_AGREEMENT` to `ACCEPT`.

      For example:

      ```
      LIC_AGREEMENT = ACCEPT
      ```
   c. Save and close the file.

What to do next

At this stage, if your intended deployment of i2 Analyze doesn't require a database, you can proceed immediately to creating it.

To learn more about the software prerequisites that i2 provides in the i2 Analyze distribution (including how to replace them in some circumstances), see Installing prerequisite software.

If your intended deployment of i2 Analyze does require a database, see Installing a database.

Installing prerequisite software

All deployments of i2 Analyze require a Java runtime, the Open Liberty application server, the Apache Solr search server, and the Apache ZooKeeper process coordinator. If you need to, you can replace the supplied version of a prerequisite by installing another supported version of the software.

Note: For more information about the system requirements and prerequisites for i2 Analyze, see the release notes.

You can find the supplied versions of all the prerequisite software in the `toolkit/bin` directory of the i2 Analyze distribution. The `wlp`, `solr`, and `zookeeper` directories each contain a single archive file, while the `java` directory has subdirectories for `linux` and `windows` distributions.

To replace any of the prerequisites before you deploy i2 Analyze, you can substitute a new archive for the supplied one. To replace one after deployment, follow the instructions in the sections below.

Java

By default, i2 Analyze uses the Eclipse Temurin JDK 17, versions of which are available for download from the Adoptium website.

To use a different distribution of Java 17, see Specifying the Java distribution. Alternatively, to use a different version of the Temurin JDK:

1. Download a supported version of the JDK from Adoptium.
2. Stop the application server and all Solr nodes.
3. Replace the archive file in the `toolkit/bin/java/linux` or `toolkit/bin/java/windows` directory with the downloaded archive.
4. Run `setup -t installJava` on all servers.
5. Start the application server and all Solr nodes.

Liberty

i2 Analyze uses an Open Liberty runtime package, versions of which are available for download from the Maven repository website.

To use a different version of Open Liberty:

1. Download a supported version of Open Liberty from Maven.
2. Stop the application server and all Solr nodes.
3. Replace the archive file in the `toolkit/bin/wlp` directory with the downloaded archive.
4. Run `setup -t installLiberty` on all servers.
5. Start the application server and all Solr nodes.

Solr

i2 Analyze uses the "slim" distribution of the Apache Solr search server. You can download the latest version from the main download page, or earlier versions from Apache's archive.

To use a different version of Apache Solr:

1. Download a supported version of the Apache Solr distribution.
2. Stop the application server and all Solr nodes.
3. Replace the archive file in the `toolkit/bin/solr` directory with the downloaded archive.

4. Run `setup -t installSolr` on all servers.
5. Start the application server and all Solr nodes.

ZooKeeper

Any new version of Apache ZooKeeper that you use with i2 Analyze must be compatible with your version of Apache Solr, *and* supported by your version of i2 Analyze. Downloads are available from the Apache ZooKeeper website.

To use a different version of Apache ZooKeeper:

1. Download a supported version of the Apache ZooKeeper distribution.
2. Stop the application server and all Solr nodes.
3. Replace the archive file in the `toolkit/bin/zookeeper` directory with the downloaded archive.
4. Run `setup -t installZookeeper` on all servers.
5. Start the application server and all Solr nodes.

Installing a database

Most deployments of i2 Analyze require a database management system. The software supports PostgreSQL, IBM Db2, and Microsoft SQL Server databases, which you must configure appropriately when you install them.

At this version of i2 Analyze, the supported versions of the database management systems are as follows:

- PostgreSQL at version 15.0 or above.

  For more information about how to install PostgreSQL, see Installing PostgreSQL for i2 Analyze.
- IBM® Db2® Enterprise Server, Advanced Enterprise Server, Workgroup Server, or Advanced Workgroup Server editions at version 11.1 Fix Pack 3 or later, or Advanced and Standard editions at version 11.5.

  Note: IBM Db2 Standard Edition - VPC Option - version 11.5 is included with i2 Analyze.

  For more information about how to install Db2, see Installing IBM Db2 for i2 Analyze.
- Microsoft® SQL Server Standard or Enterprise editions at version 14.0 (2017), 15.0 (2019), or 16.0 (2022).

  For more information about how to install SQL Server, see Installing Microsoft SQL Server for i2 Analyze.

Installing PostgreSQL for i2 Analyze

i2 Analyze supports using PostgreSQL as its database management system. This topic describes how to install PostgreSQL for use with i2 Analyze.

Location

If you are creating a production deployment, you can install PostgreSQL in any location. When you install PostgreSQL, record the location of the installation directory because you must specify this location in the deployment toolkit before you can deploy i2 Analyze.

If you are creating an **example deployment**, install PostgreSQL (server and client) in the following location:

- For Windows: `C:/Program Files/PostgreSQL/16`
- For Linux: `/usr/lib/postgresql/16`

In addition, an example deployment needs the PostgreSQL database server to listen on (the default) port `5432`.

Collation

In PostgreSQL, the collation settings that determine the case- and accent-sensitivity of comparisons in a database (for example, whether `Cafe` matches `café`) are set at creation and cannot be changed without re-creating the database.

Important: Before any deployment of i2 Analyze with the Chart Store or the Information Store, read the documentation about the `Collation_Deterministic`, `Collation_Locale`, and `Collation_Sensitivity` settings in `InfoStoreNamesPostgres.properties` and change the default values if you need to.

Other features

All installations of PostgreSQL for use with i2 Analyze must include the PostGIS (geospatial) extension.

Note: PostgreSQL packages and installers are available with and without PostGIS. For these and other downloads associated with PostgreSQL, see the official download page at https://www.postgresql.org/download.

The PostgreSQL client Command Line Tools (that is, `psql`) must be installed on the same server as the i2 Analyze toolkit.

Note: On Windows, this installation is called the "Command Line Tools". On Linux, it's just referred to as the "client".

PostgreSQL does not have a built-in scheduling agent. If you need to be able to run tasks such as deletion by rule on an automated schedule, you must install a third-party tool such as `pg_cron`.

Users

The PostgreSQL database user that you specify in `credentials.properties` must be a superuser, and it must have all available privileges enabled.

The PostgreSQL installation process creates a suitable user that has the name `postgres` and a password that you specify during the procedure.

High availability

At this version of i2 Analyze, PostgreSQL is not supported in deployments for which HADR is required.

Installing IBM Db2 for i2 Analyze

i2 Analyze supports using IBM Db2 as its database management system. This topic describes how to install Db2 for use with i2 Analyze.

Location

If you are creating a production deployment, you can install Db2 in any location. When you install Db2, record the location of the installation directory because you must specify this location in the deployment toolkit before you can deploy i2 Analyze.

If you are creating an **example deployment**, install Db2 in the following location:

- For Windows: `C:\Program Files\IBM\SQLLIB`
- For Linux: `/opt/ibm/db2/<Db2_version>`

Features and language

In all deployments, you must ensure that the following features are installed:

- Spatial Extender server support
- Spatial Extender client

For **Linux** deployments, if you are deploying with a schema that contains non-English characters, ensure that the operating system's `LANG` environment variable is set to the locale of the non-English characters. You can only deploy i2 Analyze with a Db2 instance where the `DB2_WORKLOAD` environment variable is not set and the database is row-organized. If you have an existing Db2 instance where `DB2_WORKLOAD` is set or the database is column-organized, you must create a Db2 instance with the supported configuration and deploy i2 Analyze with it. For more information about the `DB2_WORKLOAD` environment variable, see System environment. For more information about column-organized databases, see Setting the default table organization.

Users

On **Windows**, Db2 creates a Windows user account (`db2admin`), and two Windows groups (`DB2ADMNS`, `DB2USERS`). To work successfully with Db2, ensure that your Windows user account is a member of the `DB2ADMNS` >Windows group.

On **Linux**, Db2 creates an Administration Server user (`dasusr1`) and group (`dasadm1`), an instance-owning user (`db2inst1`) and group (`db2iadm1`), and a fenced user (`db2fenc1`) and group (`db2fadm1`). To work successfully with Db2, ensure that the user that runs the deployment script is a member of the `dasadm1` and `db2iadm1` groups.

Make a note of any user names and passwords that are specified during the installation process.

Note: In all scenarios, the user that you use to run the deployment scripts must have permission to create and modify the database. For more information, see CREATE DATABASE command".

Post-install

After you install Db2 for the Information Store, you must enable the administrative task scheduler and set the code page on the Db2 installation:

- On the command line, navigate to the `SQLLIB\bin` directory of your Db2 installation. On Linux, navigate to the `db2inst1/sqllib/bin` directory.
- To enable the administrative task scheduler, run the following command: `db2set DB2_ATS_ENABLE=YES`
- To set the code page for UTF-8 encoding, run the following command:

  ```
  db2set DB2CODEPAGE=1208
  ```

For more information about installing Db2, see Installing Db2 database servers.

If you installed IBM Db2 on Linux, ensure that the operating system user process resource limits (ulimits) meet the Db2 recommended values. For more information about the recommended ulimits, see Operating system user limit requirements (Linux and UNIX).

Collation

In Db2, the collation settings that determine the case- and accent-sensitivity of comparisons in a database (for example, whether `Cafe` matches `café`) are set at creation and cannot be changed without re-creating the database.

Important: Before any deployment of i2 Analyze with the Chart Store or the Information Store, read the documentation about the `Collation` setting in `InfoStoreNamesDb2Server.properties` and change the default value if you need to.

Remote Db2 database storage

If you plan to deploy i2 Analyze with remote database storage, you must install Db2 on your database server, and Db2 or IBM Data Server Client on the application server. Install Db2 according to the previous instructions; if you are using IBM Data Server Client, also ensure that Spatial Extender client support is installed. For more information about IBM Data Server Client, see Installing IBM Data Server drivers and clients.

The instance of Db2 or IBM Data Server Client on the application server must be the same version level as the instance of Db2 on the database server. For example, if the instance of Db2 on your database server is version 11.1, the instance of Db2 or IBM Data Server Client on the application server must also be version 11.1.

High availability

When you install Db2 in a deployment that uses HADR, the following must be true:

- The version of Db2 installed on the primary and standby servers must be the same.
- Install Db2 on all servers according to the previous information.
- Db2 must be installed with the same bit size (32 or 64 bit) for both the primary and standby servers.
- The primary and standby databases must have the same database name.
- The primary and standby databases must be in difference instances.
- The amount of space allocated for log files must be the same on both the primary and standby databases.

For more information about the Db2 requirements, see High availability disaster recovery (HADR) support. For more information about the Db2 HADR feature, see High availability disaster recovery (HADR).

Installing Microsoft SQL Server for i2 Analyze

i2 Analyze supports using Microsoft SQL Server as its database management system. This topic describes how to install SQL Server for use with i2 Analyze.

Location

If you are creating a production deployment, you can install SQL Server in any location. When you install SQL Server, record the location of the installation directory. You must specify this location in the deployment toolkit before you can deploy i2 Analyze.

If you are creating an example deployment, install SQL Server in the default location:

- For Windows: `C:\Program Files\Microsoft SQL Server`.
- For Linux: `/opt/mssql`. Install the SQL Server tools in the default path: `/opt/mssql-tools`.

Features

In all deployments, you must ensure that the following features are installed or enabled:

- Database Engine Services
- SQL Server Authentication
- TCP/IP Protocol

In all deployments, you must install the ODBC Driver for SQL Server and `sqlcmd` utility on your database server.

On Windows:

- Microsoft ODBC Driver for SQL Server on Windows
- `sqlcmd` Utility

On Linux:

- Download ODBC Driver for SQL Server
- Install the SQL Server command-line tools

Note: To ensure that you see drivers and tools that are compatible with your version of SQL Server, select it from the drop-down menu.

You can also install Microsoft SQL Server Management Studio to administer your SQL Server installation. If you are using SQL Server on Linux, you can install SQL Server Management Studio on a Windows workstation and connect to your SQL Server installation.

To create an example deployment of i2 Analyze on Windows, the instance name that you use must be `MSSQLSERVER`. Regardless of your operating system, the port number must be `1433`.

Users

You must have an SQL Server Authentication Login that has the following permissions:

- Server roles:
  - `dbcreator`
  - `bulkadmin`, to ingest the example data. The `bulkadmin` role is not supported on Linux
- User mappings for the `msdb` database:
  - `SQLAgentUserRole`
  - `db_datareader`

Note: In all scenarios, the user that you use to run the deployment scripts must have permission to create and modify the database.

Collation

In SQL Server, the collation settings that determine the case- and accent-sensitivity of comparisons in a database (for example, whether `Cafe` matches `café`) are set at creation and cannot be changed without re-creating the database.

Important: Before any deployment of i2 Analyze with the Chart Store or the Information Store, read the documentation about the `Collation` setting in `InfoStoreNamesSQLServer.properties` and change the default value if you need to.

Post-install

- Ensure that the `SQL Server Agent` service is running.
- On Windows, if you want to use the instance name to connect to SQL Server, ensure that the `SQL Server Browser` service is running.

Remote SQL Server database storage

If you plan to deploy i2 Analyze with remote database storage, you must install SQL Server on the database server, and SQL Server or Microsoft Command Line Utilities for SQL Server on the application server. You can install SQL Server and the Command Line Utilities according to the previous instructions.

High availability

When you install SQL Server for HADR, you must ensure that the following statements are true:

- All database servers are in the same IP address range.
- All database servers are members of the same domain for SQL Server Always On availability groups DNS name resolution.
- Two static TCP/IP addresses, one for the Windows Failover Cluster and one for the SQL Server Always On availability group. The IP addresses must be in the same range.
- The same version of Windows Server is installed.

For information about installing SQL Server with SQL Server Always On availability groups, see Prerequisites, Restrictions, and Recommendations for Always On availability groups.

Configuring i2 Analyze

During implementation of a production deployment, you need to modify the original base deployment to match the needs of your organization. When the i2 Analyze deployment is in use, you can make further configuration changes to adjust to changing needs, and to administer the system.

Configuration sections

- Configuring the i2 Analyze application
- Ingesting data into the Information Store
- Connecting to external data sources
- i2 Analyze Schema Designer

Common tasks

- Modifying a deployed i2 Analyze schema
- Configuring user security
- Connecting to external data sources
- Configuring Find Matching Records

Troubleshooting and support

- i2 Enterprise Insight Analysis support page
- i2 Support

Configuring the i2 Analyze application

To configure i2 Analyze for your organization's requirements, you complete various activities that modify its behavior. These activities can affect aspects of the system such as authenticating and authorizing users, controlling access to features, and providing appropriate search options.

If you want to modify the configuration of an i2 Analyze system that is already in production, i2 recommends that you make your changes first in the environment that is furthest away from production. For example, start by changing the configuration development environment, and then promote those changes through pre-production, test, and finally into production.

For more information about creating deployment environments, see Deployment phases and environments.

Configuring i2 Analyze schemas

An important part of developing an i2 Analyze deployment is creating the schemas for it to use. Between them, the Information Store, gateway, and connector schemas model the data that you want to analyze in the deployment.

When you deploy a schema, you also deploy one or more *charting schemes* to accompany it. These separate files specify how entities and links and their properties appear when they are visualized on charts. To provide users with more flexibility in their investigations, you can visualize data in different ways by creating multiple charting schemes.

Any schemas that you create for the i2 Connect gateway or for individual connectors are relatively flexible, and you can modify them after deployment as you see fit. However, after an Information Store schema enters production, the changes that you can make to it are strictly limited. Therefore, it is important to ensure early in the development process that if your deployment includes an Information Store, its schema meets the requirements of the organization.

Note: In some deployments of i2 Analyze, the Chart Store replaces the Information Store. The schema and the charting scheme for the Chart Store are much smaller than their equivalents for the Information Store, but changing them in a production deployment is subject to the same limitations.

You can view, edit, and create schemas with Schema Designer. For instructions on creating and editing a schema, see the i2 Analyze Schema Designer documentation.

Schema settings

When a deployment of i2 Analyze includes the i2 Connect gateway and connectors that define their own schemas, the gateway interrogates the connectors for the locations of those schemas. For schemas that are hosted on the server, the i2 Analyze application retrieves their locations from the `ApolloServerSettingsMandatory.properties` file.

For a deployment of i2 Analyze that includes the Chart Store or the Information Store, `ApolloServerSettingsMandatory.properties` must contain populated `SchemaResource` and `ChartingSchemesResource` settings that provide the locations of the schema and charting scheme files:

```
SchemaResource=
ChartingSchemesResource=
```

Note: A single deployment of i2 Analyze cannot include both the Chart Store and the Information Store. When necessary, the Information Store subsumes the Chart Store, and the Information Store schema includes the elements that describe stored Analyst's Notebook charts.

The `ApolloServerSettingsMandatory.properties` files in the example configurations that include the i2 Connect gateway contain extra settings that are named `Gateway.External.SchemaResource` and `Gateway.External.ChartingSchemesResource`. However, these names are not mandatory, and they are not the only settings for gateway schemas that can appear in the file.

In general, a deployment of i2 Analyze that includes the i2 Connect gateway can have any number of gateway schemas. Every pair of schema and charting scheme files must be identified in the `ApolloServerSettingsMandatory.properties` file, using the following syntax:

```
Gateway.ShortName.SchemaResource=
Gateway.ShortName.ChartingSchemesResource=
```

You are free to specify the `ShortName` of each pair of settings as you wish. The short name that you provide is displayed to Analyst's Notebook users in the names of item types from the schema. It is also displayed to system administrators in the user interface for creating type conversion mappings. And i2 Analyze uses the short name to differentiate between any item types that have the same name in separate gateway schemas.

Modifying a deployed Chart Store schema

A deployment of i2 Analyze that includes the Chart Store (with or without the i2 Connect gateway) supports a limited set of changes to the supplied example schema. By adding property types to the "Analyst's Notebook Chart" entity type, you can enhance users' ability to categorize and retrieve the charts that they store.

About this task

The example Chart Store schema in the deployment toolkit contains a single "Analyst's Notebook Chart" entity type. That entity type contains two property types that represent the name and description of any chart that is stored in the Chart Store. In a live deployment, i2 Analyze permits additive changes to the Chart Store schema. By adding property types to the "Analyst's Notebook Chart" entity type, you can enable users to search for charts more effectively, and to filter search results by the values of the new properties.

Note: In a deployment of i2 Analyze that includes the Information Store, the schema includes the Chart Store's "Analyst's Notebook Chart" entity type alongside all the Information Store item types. The instructions here apply to modifications to that entity type in Chart Store and Information Store schemas.

If you follow this procedure in a deployment that provides high availability, you must complete each step and run each command on every Liberty server in your environment before you move to the next step or command.

Procedure

The following steps describe how to add property types to the schema in a deployment of i2 Analyze that includes the Chart Store.

1. Locate the XML file that contains the Chart Store schema for the i2 Analyze deployment, and load it into Schema Designer.
2. Make your additions to the "Analyst's Notebook Chart" entity type, and then save the file.

   Note: Schema Designer does not validate whether your changes are compatible with the deployed schema. Validation takes place when you apply the changes to your deployment.
3. Run the following commands on the Liberty server to update the database and application to conform to the updated schema.

```
setup -t stopLiberty
setup -t updateSchema
setup -t deployLiberty
setup -t startLiberty
```

The command recognizes that you modified the schema, determines whether the changes are valid for the running Chart Store, and then applies them. If the changes are not valid, the command displays messages to explain the problems.

What to do next

By default, deployments of i2 Analyze that include the Chart Store do not specify a results configuration file. Any property types that you add are automatically available as filters for search results. However, if you add a property type that you do not intend to use for filtering, or if you have modified the "Analyst's Notebook Chart" entity type in an Information Store schema, you must set up search results filtering to include or exclude the new property types as required.

When you complete the procedure, reconnect to i2 Analyze from Analyst's Notebook to confirm that the changes are present and behaving correctly.

Modifying a deployed Information Store schema

i2 Analyze supports a limited set of small changes to the Information Store schema in a production deployment. In general, if your changes only add to the existing schema, you can apply them without the disruption that more significant changes can cause.

About this task

After you deploy i2 Analyze with an Information Store, you can generally make additive changes to the Information Store schema, but not destructive ones. For example, you can add new item types, and add new property types to existing item types, but you cannot remove types from the schema. For more information about the changes that you can make, see Permitted Information Store schema changes.

Destructive schema changes include removing item types or property types. Removing types can result in an Information Store with data that is not valid according to the schema. To make destructive changes to the schema, you must remove any data from the system and re-create its databases. For more information about performing destructive schema changes or replacing the schema, see Replacing the Chart Store or Information Store schema.

If you follow this procedure in a deployment that provides high availability, you must complete each step and run each command on every Liberty server in your environment before you move to the next step or command.

Procedure

The following steps describe how to make small changes to the schema in a production deployment of i2 Analyze that includes an Information Store.

1. Locate the XML file that contains the Information Store schema for the i2 Analyze deployment, and load it into Schema Designer.
2. Make your changes to the schema and its associated charting schemes, and then save the file.

   Note: Schema Designer does not validate whether your changes are compatible with the deployed schema. Validation takes place when you apply the changes to your deployment.
3. Run the following commands on the Liberty server to update the database and application to conform to the updated schema.

```
setup -t stopLiberty
setup -t updateSchema
setup -t deployLiberty
setup -t startLiberty
```

The command recognizes that you modified the schema, determines whether the changes are valid for the running Information Store, and then applies them. If the changes are not valid, the command displays messages to explain the problems.

Note: If you customized the Information Store creation process by specifying `create-database="false"` in the topology file and running the scripts yourself, this command works in the same way. Execution stops so that you can customize the changes to the Information Store. After you apply the changes, you can run the task again to complete the process.

What to do next

Because you can make only additive changes to an Information Store schema that you modify through this procedure, it is not mandatory to change other parts of your deployment. However, to take full advantage of your additions, consider the following complementary changes.

- To enable the Information Store to ingest data for the new item types and property types, modify your ingestion artifacts. See Information Store data ingestion.
- If your deployment includes definitions of the property values of merged i2 Analyze records, you must update your merged property values definition views. See Define how property values of merged records are calculated.
- If you want users to see the new types in quick search filters, edit the configuration file that controls them. See Setting up search results filtering.
- If your deployment includes highlight queries, you can update them for the new item and property types. For more information, see Deploying highlight queries.

Permitted Information Store schema changes

After you create an Information Store schema and use it in a deployment, i2 Analyze restricts subsequent changes to that schema to ensure that data in the system remains accessible. In general, you can add content to a published Information Store schema, but you cannot take content away.

To be specific, i2 Analyze prevents changes to a schema that might invalidate data that is already stored. For example, if you remove an item type from the Information Store schema, then the store might contain data for which i2 Analyze no longer has a definition. However, you can make additive changes to the schema. You can also disable or hide item or property types in some parts of the system, if you are certain that they are no longer required.

Permitted changes

You can make the following changes to the Information Store schema of a live deployment:

- Add an item type.
- Change the display name of an item type.
- Change the icon of an item type.
- Add a property type.
- Change the display name of a property type.
- Change the display order of a property type.

- Increase the value length of a property type.
- Make a property type non-mandatory.
- Add a grade type.
- Add a labeling scheme.

Prevented changes

i2 Analyze prevents all of the following Information Store schema changes from taking place against a live deployment:

- Change the schema identifier.
- Remove an item type.
- Remove an entity type from the permitted list for a link type.
- Remove a property type.
- Make a property type mandatory.
- Remove a default property value.
- Remove a property value from a selected-from or suggested-from list.
- Change the logical type of a property type.
- Remove a grade type.
- Remove a link strength.

To protect your data when you redeploy with a modified Information Store schema, i2 Analyze carries out validation checks to ensure that the changes you made do not result in data loss.

Permitted changes that do not affect the Information Store

You can make the following changes to the Information Store schema of a live deployment. However, they do not affect the store, which means that users still see items with hidden types when they run a quick search, for example:

- Add a link constraint.
- Add a link strength.
- Disable an item type.
- Hide an item type.

The disable and hide functions change how item and property types behave in the deployment without making any destructive changes. Use these features with caution though because they can affect the behavior of visual query and import operations.

Replacing the Chart Store or Information Store schema

The schema is a key component of any i2 Analyze deployment that includes the Chart Store or the Information Store, and configuring it correctly is an important part of the development process. Replacing or making destructive changes to the schema requires you to clear and repopulate the store. As you do so, you might need to change other parts of the system too.

About this task

This procedure contains the steps for making destructive changes to a Chart Store or Information Store schema in your deployment of i2 Analyze. When you replace the schema, you must clear data from

the system, modify any sample data, and edit any other part of the configuration that depends on the schema.

In a deployment that provides high availability, use the documentation from your database management system to remove the Chart Store or Information Store database from each database server in your deployment instead of the `dropDatabases` toolkit task.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `deleteSolrCollections` and `createSolrCollections` on one Liberty server only.

Procedure

1. Copy the new schema and charting scheme files to the `configuration\fragments\common \WEB-INF\classes` directory of the deployment toolkit.
2. Specify the schema and charting scheme that the deployment uses.
   a. Using a text editor, open the `ApolloServerSettingsMandatory.properties` file in the same directory as the schema files.
   b. Set the values of the `SchemaResource` and `ChartingSchemesResource` properties to the names of your schema and charting scheme files.
   c. Save and close the file.

The following steps update your deployment with the new schema.

1. Stop the deployment:

   ```
   setup -t stopLiberty
   ```
2. To remove the database and Solr collections, navigate to the `toolkit\scripts` directory and run the following command:

   ```
   setup -t dropDatabases --hostname <liberty.host-name>
   setup -t deleteSolrCollections --hostname <liberty.host-name>
   ```

   Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

   A message is displayed when you run each task to confirm that you want to complete the action. Enter `Y` to continue. The database and Solr collections are removed from the system.
3. To re-create the Solr collections and databases, run the following commands:

   ```
   setup -t createSolrCollections --hostname <liberty.host-name>
   setup -t createDatabases
   ```
4. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```
5. Update the schema:

   ```
   setup -t updateSchema
   ```
6. Start Liberty:

   ```
   setup -t startLiberty
   ```

What to do next

If you are using a results configuration file, configure the facets to match the item and property types that you added or changed in the Information Store schema. See Setting up search results filtering.

If you are defining the property values of merged i2 Analyze records, you must update your merged property values definition views. See Define how property values of merged records are calculated.

Adding, removing, modifying gateway schemas

A single deployment of i2 Analyze that includes the i2 Connect gateway can contain several gateway schemas. When you add or remove gateway schemas, you must edit the `ApolloServerSettingsMandatory.properties` file. It is likely that you must edit some of the other configuration files and settings too.

About this task

When several connectors query the same data source, it can make sense for them to share a gateway schema instead of defining schemas of their own. If you buy or develop more than one set of such connectors, then it is possible for your deployment of i2 Analyze to use several gateway schemas. The section of the settings file then looks like this:

```
Gateway.<SourceOne>.SchemaResource=
Gateway.<SourceOne>.ChartingSchemesResource=
Gateway.<SourceTwo>.SchemaResource=
Gateway.<SourceTwo>.ChartingSchemesResource=
```

For each gateway schema that you add to a deployment, you must add two entries to this list with a new name in place of `<SourceOne>` or `<SourceTwo>`. If you remove a gateway schema, make sure to remove both entries that were associated with that schema.

If you follow this procedure in a deployment that provides high availability, you must complete each step and run each command on every Liberty server in your environment before you move to the next step or command.

Procedure

After you add or remove the settings for a particular gateway schema from `ApolloServerSettingsMandatory.properties`, check other aspects of your system configuration.

1. Before you reload the i2 Analyze server to reflect your changes, make sure that the results configuration file matches your new set of gateway schemas.
   a. Open the file indicated by the `ResultsConfigurationResource` property in `DiscoServerSettingsCommon.properties` and follow the instructions in Setting up search results filtering to remove references to item types from removed gateway schemas.
   b. Add or edit elements in the file to enable result filtering on item and property types in any new gateway schemas.
2. Use the i2 Analyze server admin console to reload the server and update its information about the gateway schemas.
3. Follow the instructions in Configuring matching to make any changes to the system or Find Matching Records match rules files that are required as a result of your changes to the configured schemas.
4. If your deployment uses them, update the set of type conversion mappings.
   a. Open a web browser and navigate to the i2 Analyze server admin console.
   b. In the **i2 Analyze type conversion** app, add, edit, or remove type conversion mappings to reflect the new set of gateway schemas.

c. Export the modified conversion mappings. Copy the resulting `mapping-configuration.json` file to the deployment toolkit's `configuration\fragments\common\WEB-INF\classes` directory.

d. Redeploy Liberty to update the type conversion mappings on the server.

Results

On completion of the above steps, your i2 Analyze deployment is fully updated in response to the changes that you introduced by adding, removing, or modifying a gateway schema.

i2 Analyze Schema Designer

In i2 Analyze deployments, schemas are key to how data is visualized, analyzed, and stored. i2 Analyze Schema Designer is a Windows application for creating and editing schemas that meet all of the requirements i2 Analyze places upon them.

i2 Analyze schemas, and the charting schemes that accompany them, are stored as XML files. Using Schema Designer to create and edit schemas has several benefits:

- Schema files that are guaranteed to be well formatted and valid for use with i2 Analyze
- Automatic creation and maintenance of identifiers and other system information that schemas require
- Built-in access to the i2 Semantic Type Library, which unlocks advanced features for search and analysis
- The ability to create and edit charting and labeling schemes alongside the schema itself

This documentation describes how to use i2 Analyze Schema Designer as part of the process for designing, creating, deploying, and modifying i2 Analyze schemas.

Important: Schema Designer includes features for maintaining schemas that target earlier versions of i2 Analyze. This documentation does not describe how to use these features.

Designing schemas

The schemas in an i2 Analyze deployment describe how the data in your investigations is expressed as the entities, links, and properties of the i2 Analyze data model. Before you start to use Schema Designer, you should have a clear idea of the schema that you want to create or update.

Whether you're creating a schema or editing an existing one, you need to understand both the i2 data model and the data you're working with. If you're developing a new schema, Schema Designer becomes part of the iterative development process. If you're modifying a schema, Schema Designer helps you to retain consistency as you make your changes.

Data modeling

The detail of how you model data for your organization can be different depending on whether your schema is for an Information Store or an external data source:

- For an Information Store schema, you often have a body of existing data to work with, as well as analysts who are familiar with it. These schemas can be quite large, describing most or all of the data in the organization.
- For a connector (or a gateway) schema, the schema is tied closely to one or more connectors, and it's often developed in parallel with them. These schemas are generally smaller, describing only the data that the connector can return.

In either case, Schema Designer presents and allows you to manipulate schemas in terms of the i2 Analyze data model. For an example of understanding data and then describing it in terms of entity, link, and property types, see Designing an i2 Analyze connector schema in the i2 Connect SDK.

Schema development

If your deployment of i2 Analyze includes an Information Store, its schema is a fundamental part of that deployment. The Information Store's schema affects the structure of its data storage, and as a result it's difficult to make changes after the deployment enters production.

When you create an Information Store schema, you can employ a schema development environment that allows you to iterate and test your design before you commit to its final form.

Gateway and connector schemas are not constrained in the same way, but a schema development environment can still be a useful way to iterate quickly during schema development.

If Analyst's Notebook users will connect to your i2 Analyze deployment, then as your schema matures you should begin to develop the charting schemes that determine how data from i2 Analyze is represented by the items on Analyst's Notebook charts.

Schema maintenance

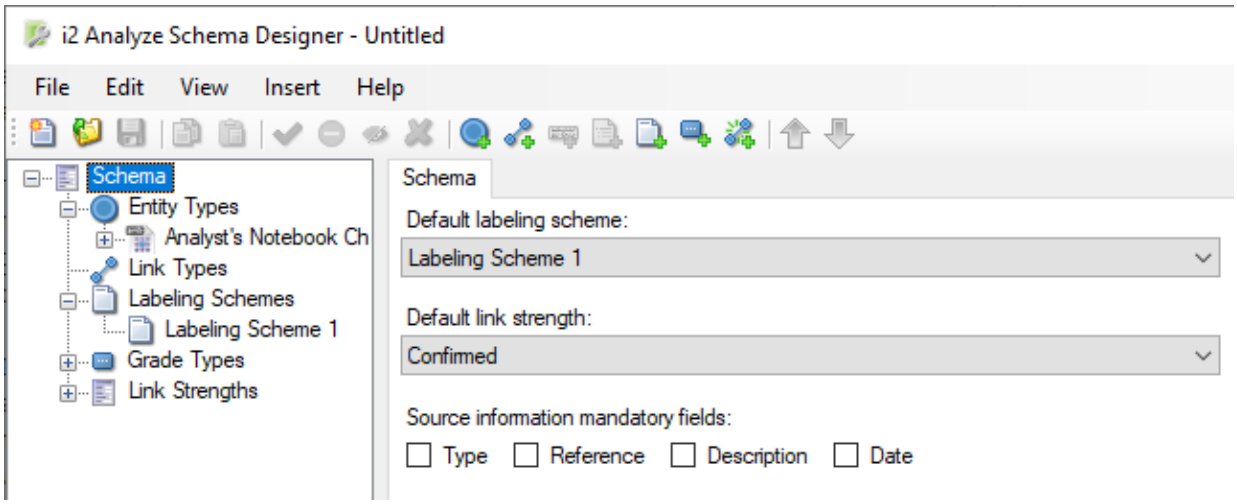
As suggested above, the changes that you can make to a schema after you've deployed it to users depend on whether it's for an Information Store, an i2 Connect gateway, or a connector to an external data source. Schema Designer validates your edits to ensure consistency, while i2 Analyze itself determines whether the changes are permitted in context.

For more information about adding, modifying, and replacing the schemas in a live i2 Analyze deployment, see the configuration documentation.

Creating a schema

When you have an initial plan for the entity, link, and property types that your deployment of i2 Analyze will use, you can start to create the schema in Schema Designer. Usually, it's better to add entity types to the schema before you add link types.

To create a schema, just start i2 Analyze Schema Designer. The application window opens to display a new, empty schema that's ready for you to work with.



Note: The tree view presents the possible contents of the schema as entity types, link types, labeling schemas, grade types, and link strengths. Grade types and link strengths have no effect in recent versions of i2 Analyze.

Alternatively, for a demonstration of what a complete i2 Analyze schema can look like, open any of the example schemas that i2 provides.

Important: All schemas that you create through Schema Designer contain system types that must be present for the schema to function correctly. The application doesn't allow you to remove these types, or to change them in a way that makes the schema invalid.

Creating entity types

Entity types are the templates that describe how entities - the real-world objects that your data represents - are stored, analyzed, and visualized in i2 applications. When you create entity types, provide as much information as you can for the developers and analysts who will use them.

About this task

The process of creating an entity type involves not only naming and describing the type itself, but also choosing the semantic type that categorizes it and creating the property types that it contains.

Procedure

To create an entity type in Schema Designer:

1. In the application window, click **Insert > New Entity Type**.

   A **New Entity Type** is added to the list of Entity Types in the navigation tree. The Entity Type tab displays **Name** and **Description** fields that you can edit, as well as a default icon and an indicator that a semantic type is yet to be assigned.
2. Provide a name and a description for the new entity type, and select the icon that entities with this type have by default in visualizations.
3. Next to the **Semantic Type** field, click **Select** to choose a semantic type for this entity type.
4. Create the property types that define the data values that entities with this type can contain.

Results

The entity type is successfully created, complete with your semantic and property type specifications.

Creating link types

Link types are the templates that describe how links - which represent associations between entities - are stored and analyzed in i2 applications. When you create link types, take care to ensure that your end type selections match your schema design.

About this task

The process of creating a link type involves not only naming and describing the type itself, but also choosing the semantic type that categorizes it and creating the property types that it contains.

In addition, creating a link type means specifying which types of entities can appear at the ends of links of that type. For example, a link of type Owner might represent associations between entities of type Person, Organization, Property, and Vehicle.

Procedure

To create a link type in Schema Designer:

1. In the application window, click **Insert > New Link Type**.

   A **New Link Type** is added to the list of Link Types in the navigation tree. The Link Type tab displays **Name** and **Description** fields that you can edit, as well as an indicator that a semantic type is yet to be assigned.
2. Provide a name and a description for the new link type.
3. Next to the **Semantic Type** field, click **Select** to choose a semantic type for this link type.
4. Specify the types of entities that can appear at the "to" and "from" ends of links with this type:
   a. Switch to the Link Ends tab. The **From End Types** and **To End Types** lists present all the entity types in the schema as potentially valid link ends.

      Note: This behavior is why it's helpful to create all the entity types before you create the link types.
   b. Select the types of entity that are valid ends for links of this type. If it's valid for an entity type to appear at both ends, you must select it in both lists.
5. Create the property types that define the data values that links with this type can contain.

Results

The link type is successfully created, complete with your semantic and property type definitions and its link end constraints.

Creating property types

The property types that entity and link types contain are the templates for the properties that entities and links contain. When you create property types, you provide the basis on which i2 applications perform comparisons and analysis of your data.

About this task

The process of creating a property type involves not only naming and describing it, but also setting parameters for the values that properties with the type can have.

For example, all property types have a logical type that restricts what kind of values their properties have. But for some logical types, you can also place constraints on the size of those values, or even limit them to a predefined set.

Procedure

To create a property type in Schema Designer:

1. In the navigation tree, select the entity or link type to which you want to add the new property type.
2. Click **Insert > New Property Type**.

   A **New Property Type** is added to the list of property types for the entity or link that you selected. The Property Type tab displays **Name** and **Description** fields that you can edit, as well as an indicator that a semantic type is yet to be assigned.

   The tab also contains the fields that govern the values of properties with this type.
3. Provide a name and a description for the new property type. The name has no enforced restrictions, but keep in mind that it will appear in column headings and similar locations. Shortness can therefore be an advantage.
4. Next to the **Semantic Type** field, click **Select** to choose a semantic type for this property type.
5. Use the **Logical Type** drop-down to specify the logical type that values of properties with this type must have.
   - If you choose a string type, you can also specify a **Maximum Length** for property values.
   - If you choose a selected-from or suggested-from list type, you should additionally create the values that users see by adding them to the **Selection List**.
6. To specify that a property of this type must have a value in all entities or links that contain it, select **Is Mandatory**.

Results

The property type is successfully created, complete with your semantic and logical type specifications. You can now use this type in labeling schemes and charting schemes.

Specifying logical types

The logical type of a property type controls what kind of value a property with that type can store, and it affects the searching, matching, and comparison operations that are available to users. Logical types also affect the validation that takes place when data is entered or imported.

About this task

In Schema Designer, the **Logical Type** drop-down on the Property Type tab contains the following logical types that i2 Analyze supports:

- Boolean
- Date
- Date & time
- Decimal
- Double
- Geospatial
- Integer
- Multiple-line string
- Selected-from list
- Single-line string
- Suggested-from list
- Time

Note: The drop-down also contains types named Document, Picture, and XML. These types are not valid in schemas for recent versions of i2 Analyze.

String types

In an i2 Analyze schema, you can specify whether the string values of a property can be "single-line" or "multiple-line". Your choice should be based on the intended use of the data, and the performance of your system.

A single-line string property is used to store text that can be used for filtering, and supports operations such as 'Starts with' and 'Ends with'. Conversely, a multi-line string property can store larger blocks of information, but does not support those operations.

In general, you should try to use single-line strings for values of up to a few hundred characters, and multi-line strings for bigger values.

Procedure

To specify a logical type for a property type in Schema Designer:

1. In the Property Type tab, select a logical type from the **Logical Type** list. The default logical type is Single-line string.
2. If you selected either Selected-from list or Suggested-from list, provide values by adding entries to the **Selection List**. You can add entries manually by clicking **Add**, or from a file by clicking **Import**.

   The file that contains values must be in plain text format, with the following structure:

   ```
   <value> <tab> <description>
   <value> <tab> <description>
   ```

   Note: The importer ignores the first line of the file, which you can therefore use for headings or a comment, for example.
3. **Optional:** For all string values (single-line, multiple-line, and selected-from and suggested-from lists) you can specify a **Maximum Length**.

   Note: To aid performance, the default maximum length for single-line strings is 250 characters. Increasing this limit can affect performance and must be tested.

Creating labeling schemes

In i2 applications, the label is usually the first thing that users see when an entity or a link of a particular type is displayed. In an i2 Analyze schema, a labeling scheme defines how labels are to be constructed from property values.

About this task

Each labeling scheme in a schema can contain instructions for assembling labels for all the entity and link types in that schema. For each type, you can specify the combination of property values and free text that will make up the label.

For example, for an entity of a type named Person, a reasonable label might be the first and last name of that person. You might specify that the First Name and Last Name property types of the Person type should form the parts of its label.

Note: To maintain consistency, you can configure a charting scheme to use a labeling scheme to determine the labels of chart items.

Procedure

All new schemas in Schema Designer start with a labeling scheme named **Labeling Scheme 1**. You can either edit the default scheme or create one by clicking **Insert > New Labeling Scheme**.

Important: The i2 Analyze server uses only the *default* labeling scheme in the schema to generate the labels for entities and links, and you should prioritize it. Any other labeling schemes can be used only in charting schemes.

With the labeling scheme that you want to edit selected in the navigation tree, you can go ahead and create its label definitions:

1. In the Labeling Scheme tab, select the entity or link type whose label you want to define in the **Item type** list.

   The **Properties** list is updated with the property types that your chosen type contains.
2. Select the property types to use in the label, and click the arrow to move them to the **Label parts** column.
3. **Optional:** Create label parts that contain fixed text by entering them in the **Free Text** box and clicking the corresponding arrow.

   You can use free text to separate combined values in a label. For example, you might put a space between the First Name and Last Name in a label for the Person type. You can also use free text to provide a standard label for entities or links that don't need value-based labels.

   Note: For an entity or link type with an empty **Label parts** list, the system displays **<No label>** for an entity or link of that type. Adding a free text label part containing any character (other than a space) changes the behavior so that the system displays the text instead.
4. **Optional:** Rearrange the label parts in the list by selecting them and clicking the up and down arrows as required.

Results

After you complete the procedure for all of the entity and link types in your schema, the labeling scheme is ready for use by the i2 Analyze server, or in the charting schemes that Analyst's Notebook uses.

Editing a schema

While you edit your schema, it must pass the Schema Designer validation checks. These checks do not prevent you from editing your schema, but make you aware of any invalid changes that are made.

These validation checks ensure that your schema contains all the necessary components for it to function, such as link end constraints. Any issues that are found in the schema can be found at the bottom of the application window for you to fix.

Charting schemes

A charting scheme is a collection of mappings from the properties of i2 Analyze records to the properties and attributes of Analyst's Notebook chart items. Charting schemes can affect both the appearance and the behavior of Analyst's Notebook chart items that contain i2 Analyze records.

If you want to change how item labels are constructed from record data, or to enable displaying attributes on the chart whose content comes from record data, or to provide item date and time values from record data so that users can perform analysis in timeline charts, then you need to create a charting scheme.

Creating charting schemes

When you create a schema in Schema Designer, the application automatically creates four charting schemes, named Charting Scheme 1 thru 4. Each charting scheme contains entries for all the entity and link types in the schema, as well as default entries whose mappings are used when the charting scheme has no more specific mappings.

However, the entries in the automatically created schemes are all empty. To produce a charting scheme that has an effect, you must either edit one of them, or configure a new one of your own.

Before you begin

Analyst's Notebook chart items have their own set of properties that are distinct from the properties of i2 Analyze records. By default, when a chart item contains an i2 Analyze record, the property values from that record are visible in Record Inspector but not in the **Edit Item Properties** dialog. Analyst's Notebook populates the item label from the record label, but the attributes and the date and time properties are empty.

The purpose of a charting scheme is to map from property types in the schema to the properties and attributes of Analyst's Notebook items. You can arrange for users to be able to switch between behaviors by creating multiple charting schemes. The effectiveness of a particular charting scheme depends partly on your schema and its labeling scheme. Ideally, your schema should be mostly complete before you start to create charting schemes for it.

Procedure

To create or edit a charting scheme:

1. In the application window, click **File > Edit Charting Schemes**.

   If your current schema has no associated charting schemes, a dialog window opens with the four empty charting schemes in a navigation tree. If your schema does have associated charting schemes, the navigation tree displays them.

   Note: The **Create Data Records** setting has no effect on charting schemes for recent versions of i2 Analyze.

2. **Optional:** To base a new charting scheme on one that you already have, select the existing charting scheme in the navigation tree and click **Duplicate**. The new charting scheme appears in the navigation tree with the prefix "Copy of".
3. Create, edit, and delete charting schemes as you need. In general, try to delete schemes that you don't want, and rename the charting schemes that you want to keep.

   Important: Changes that you make to charting schemes in the Edit Charting Schemes dialog window (and any of its children) are not retained until you click **OK** to close the dialog. To save the changes, you must save the schema itself.

Editing charting scheme property mappings

In the Edit Charting Schemes window, each charting scheme in the navigation tree contains a list of the entity and link types that the schema defines. To map from a schema property type to a property of an Analyst's Notebook chart item, you add an entry for the chart item property to the list in the relevant entity or link type.

For example, to control the label of an Analyst's Notebook chart item that contains an i2 Analyze entity record of type Person, you add the **Label** chart item property to the **Properties** list for the Person entity type. You can also edit the Default entity (or link) type to control how chart item properties are populated when the charting scheme doesn't contain configuration for a particular type.

Before you begin

The full list of chart item properties that you can populate through the mappings in a charting scheme is as follows:

- Label
- Date
- Time
- Description of date & time
- Description
- Source reference
- Source type

When you edit the mapping for a chart item property, you create an ordered list of schema property types and other sources whose values are eventually combined to form the chart item property value. You also have the option to add a prefix or a suffix (often a space or some other punctuation) to the value from each source in the list.

The label for an entity or link type that you defined in the labeling scheme is available for use in all chart item properties for that type in the charting scheme, but it's typical to use it specifically for the **Label** property.

Default labels for links

A common application for a charting scheme is to provide a label that uses some fixed text for chart links, when those links contain i2 Analyze records for which the charting scheme has no explicit label configuration.

Dates and times for timeline analysis

The Date and Time chart item properties are essential to creating and using timeline charts in Analyst's Notebook. To participate in a timeline chart, a chart item must have values for its Date and Time properties. You can provide those values by use a charting scheme to map the values of i2 Analyze record properties to them.

Procedure

To map a schema property type to a chart item property:

1. In Schema Designer, open the i2 Analyze schema whose charting schemes you want to edit.
2. From the application window, click **File > Edit Charting Schemes** to open the Edit Charting Schemes window.
3. Expand the charting scheme that you want to edit, and then find the entity or link type from the schema that you want to create a mapping for. (Alternatively, select the **Default** type in either category.)
4. Right-click the **Properties** entry under your selected entity or link type, and then click **Insert Chart Item Property Type**.

   The **Add Chart Item Property Type** dialog appears.
5. From the list in the dialog, select the chart item property that you want to map to and click **OK**. The list of available properties gets shorter as you add more mappings.

   Schema Designer displays a list of the sources whose values will populate the chart item property. For a new mapping, the list starts empty. You can use the **Insert** button to add sources to the list.
6. Add sources to the list, and if necessary put them in the order that you want them to appear in chart item property values.
7. Click **OK** to confirm your changes, save the schema, and then redeploy it to the i2 Analyze server.

Results

When the mapping is complete, the items on an Analyst's Notebook chart that uses this charting scheme will have their properties populated from data in the i2 Analyze records that they contain.

Editing charting scheme attribute mappings

You can use a charting scheme to map from schema property types to the attributes of Analyst's Notebook chart items, as well as to their properties. Charting schemes can contain definitions of new attribute classes whose instances you can add to Attributes lists in the Edit Charting Schemes window.

For example, to arrange for Analyst's Notebook chart items to display the dates of birth of the Person entity records that they contain, you add an instance of a Date & Time attribute class to the Attributes list for the Person entity type.

About this task

When you first start to map schema property types (and other sources) to attribute instances, you'll need to create the attribute classes that define those instances at the same time. As you build up the charting scheme, it's likely that you can reuse the classes to add attribute instances to the lists of multiple entity types.

After you add an attribute to an entity or a link type in the charting scheme, the process for specifying the value of the attribute is similar to the process for specifying chart item property values.

Procedure

To map a schema property type to a chart item attribute:

1. From the application window, click **File > Edit Charting Schemes** to open the Edit Charting Schemes window.

2. Expand the charting scheme that you want to edit, and then find the entity or link type from the schema that you want to create a mapping for. (Alternatively, select the **Default** type in either category.)
3. Right-click the **Attributes** entry under your selected entity or link type, and then click **Insert Attribute Instance**.

   The **Insert an Attribute Class** dialog appears. You can choose to create an instance of an attribute class that already exists in the charting scheme, or to select a schema property type (or a similar source) whose value you want to create an attribute class around.
4. To create an instance of an existing attribute class:
   a. Click **Select an existing attribute class**, and choose an attribute class from the list.

      Note: The list of available classes is filtered by the logical types of the property types in the schema. For example, if your schema contains no numeric property types, you can't use an attribute class of type Number.
   b. Click **OK** to add an instance of the attribute class to the Attributes list. Schema Designer then provides a way to specify values for the attribute, the details of which depend on the type of the attribute.
5. To base a new attribute class on a schema property type:
   a. Click **Select a repository property**, and then select the type of the property whose values you want to use in attributes from the list.
   b. If you want the new attribute class to have a different name from the property type you selected, edit the **Specify the new attribute class name** field.
   c. Click **Next**, and then configure the appearance and behavior of instances of the new attribute class.
   d. Click **OK** to create an attribute class whose type is consistent with the property type that you selected, *and* add an instance of that class to the Attributes list.
   e. Edit the new class and its instance to perform any further customization.

Results

When the mapping is complete, the items on an Analyst's Notebook chart that uses this charting scheme will have attributes whose values are populated from data in the i2 Analyze records that they contain.

Creating attribute classes

In Schema Designer, as well as basing a new attribute class on a schema property type, you can also create attribute classes from scratch. Any attribute class that you create in this way can be used to add attributes to entity and link types throughout its parent charting scheme.

About this task

The configurable options for all attribute classes include their name, their semantic type, and the representation of their instances on Analyst's Notebook charts. You can also control how attributes behave when chart items of the same type are pasted or merged together.

Procedure

To create an attribute class for use in a charting scheme:

1. In the Edit Charting Schemes window, expand the navigation tree for the charting scheme you want to add the class to, and right-click **Attribute Classes**.
2. Select the type of attribute class that you want to create: **Text**, **Flag**, **Number**, or **Date & Time**.

   The new class is added to the charting scheme.
3. Configure the details of the new attribute class, and then click **OK** to save the charting scheme, or continue by using it to create attribute instances.

Results

The new attribute class is ready for use creating attribute instances in this charting scheme.

Configuring link summarization

On an Analyst's Notebook chart, two chart entities can be connected by several links that contain i2 Analyze records of the same type. Through your charting scheme settings, you can control how Analyst's Notebook represents such links.

About this task

Schema Designer provides the options for displaying multiple links of the same type between the same two entities in its **Link Summarization** settings. For each type, the available behaviors are as follows:

Option	Description
Single Link	Chart links containing records of the same link type are combined into a single link. This option is useful if you are producing a summary chart and do not want to show all the details.
Directed	Chart links containing records of the same link type *in the same direction* are combined. This option is useful when charting information such as telephone calls or financial transactions.
Multiple	Chart links containing records of the same link type are charted separately. This option is useful if you want to show all the detail, provided that the chart doesn't become hard to read.
Flow	Chart links containing records of the same link type are combined into a single link whose direction is determined by the values of properties whose type you specify. For example, if there are several financial transactions between two bank accounts, this option is useful for representing the aggregate flow of capital.

Note: The options that refer to the strength of links have no effect on charting schemes for recent versions of i2 Analyze.

Procedure

To configure link summarization for a link type:

1. In the Edit Charting Schemes window, find the charting scheme that you want to edit in the navigation tree, and view its **Link Types**.
2. Select the link type that you want to configure link summarization for. The **Link Summarization** settings appear to the right of the tree.
3. Select one of the summarization options, as described above.

   Note: The **Flow** option is available only when the selected link type has a numeric data property. If it has several such properties, you can select the property type that you want to use.

   If you select the **Flow** option, the aggregate flow value is calculated and becomes available for you to use in property or attribute mappings.

Results

In an Analyst's Notebook chart that uses this charting scheme, chart links containing i2 Analyze records of your selected type behave as directed when they connect the same two chart entities.

Copying mappings between charting schemes

At any level in the structure of a charting scheme, you can copy the definitions that it contains to another charting scheme. For example, you might want to reuse the all attribute classes that you've defined; or you might just want to copy a single property mapping between schemes.

Procedure

To copy any element from one charting scheme to another:

1. In the Edit Charting Schemes window, use the navigation tree to select the element that you want to copy. Schema Designer supports copying *any* element beneath the top-level, charting scheme elements.
2. Click **Copy to** to display the **Copy To** window, which contains a list of the charting schemes to which you can copy your definitions.
3. Select the charting schemes that you want to copy the element to, and click **OK**.

   Note: The **Copy To** window indicates whether a definition exists in the target scheme for the entry you want to copy. Any existing definition in the selected charting scheme is overwritten.

Results

The definitions in the selected entry, along with all its child entries, are copied to the selected charting scheme. For example, if you select Entity Types in the navigation tree, all of the property and attribute mappings for all of the entity types are copied.

Charting scheme validation

Schema Designer validates charting schemes when you create them, while you edit them, and when you load the schema to which they belong. If a charting scheme is modified outside the application, opening its parent schema provides a way to check its validity.

When you open a charting scheme in Schema Designer, the following validation checks occur:

- Whether the charting scheme's underlying XML is well-formed and valid.

- Whether the entity types and link types in the charting scheme match the type definitions in the schema. If some types don't match, Schema Designer displays an error log and prevents the charting scheme from opening until the errors are corrected.
- Whether the logical types of property types that the charting scheme relies on match the logical types that are specified in the schema. If some types don't match, then Schema Designer again displays an error log and prevents the charting schema from opening.

Assigning semantic types

Assigning a semantic type to the entity, link, and property types in your schema is a way of further categorizing and identifying relationships within your data. i2 applications can use semantic information to infer that two things are *like* each other, even if they don't have the same entity, link, or property type.

About this task

For some forms of analysis, i2 software interprets entities, links, and properties based on their semantic types:

- You might assign the Motor Vehicle semantic type to a Car entity type and a Bus entity type. If you search for motor vehicles on a chart, Analyst's Notebook can find entities of both types.
- In the Semantic Type Library, Passport Number and Social Security Number are both children of the National Identifier semantic type. An external search that requires seeds with National Identifier properties will accept any entity with a property that has any of its child semantic types.

Note: In an i2 Analyze deployment that uses several schemas, such as one that retrieves data from external sources, assigning appropriate semantic types is especially important. Entity, link, and property types don't necessarily match across schemas, but semantic types do.

Procedure

To assign a semantic type to your entity, link, or property type:

1. In the Schema Designer application window, select the type that you want to assign a semantic type to.
2. In the type on the right of the window, click **Select** next to the **Semantic Type** field to open the **Select Semantic Type** window.

   The window provides a view of the i2 Semantic Type Library, which contains the semantic types that are shared by all i2 applications.
3. Browse the library to find and select the semantic type that categorizes your data as specifically as possible.

   Note: If you can't find an appropriate semantic type in the library, consider creating a custom semantic type.
4. Click **Assign** to set the semantic type and close the window.

Deriving custom semantic types

If the i2 Semantic Type Library does not contain a semantic type that describes an entity, link, or property type in your schema, Schema Designer offers the ability to create one that does. You can derive a custom semantic type from an appropriate parent semantic type.

About this task

Caution: Defining a custom semantic type when the library already contains an appropriate semantic type can cause problems for data analysis. Search carefully for an available semantic type before you create your own.

When you create a custom semantic type, it's important to choose an appropriate parent type. Analyst's Notebook and other i2 applications rely on parent semantic types to perform generalized analysis correctly.

Procedure

To derive a custom semantic type:

1. In the **Select Semantic Type** window, use the **Find** field to locate an existing semantic type that's similar to the one you require. The search uses the names, synonyms, and descriptions of the types in the library.
2. Select the type that you want to be the parent of your new type, and then click **Derive Custom Type**. A new semantic type is added as a child of the selected type.

   Note: Custom semantic types have a red icon that differentiates them from the standard types in the i2 Semantic Type Library.
3. Change the name of the custom semantic type to one that reflects your intended use.
4. To improve results when search for this semantic type in the future, add synonyms for it in the **Synonyms** field. Separate each synonym with a comma. Phrases are allowed.
5. Describe your intended use for the custom semantic type in the **Description** box. Optionally, you can include the URL of a web page that provides more information.
6. Click **Assign** to complete the definition of the custom semantic type and assign it to the entity, link, or property type that you started from.

Results

As well as being assigned to the entity, link, or property type, the new custom semantic type is added to the semantic template library in the schema. Optionally, you can use the commands under **File > Custom Semantic Types** to save the library and then merge it into another schema, bringing your custom semantic types with it.

Glossary

This glossary provides terms and definitions for the i2 Analyze software and products.

The following cross-references are used in this glossary:

- *See* refers you from a non-preferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

A

abstract semantic type

A semantic type that only serves as the parent of other semantic types. Abstract semantic types categorize their child semantic types, but are never associated with real data.

access level

A measure of the rights that a user has to view or edit an item. Access levels are calculated separately for every user and every item. See also grant level, security dimension.

administrator

A person responsible for administrative tasks such as access authorization and content management. Administrators can also grant levels of authority to users.

alert

A message or other indication that signals an event or an impending event that meets a set of specified criteria.

alert definition

The statement of criteria that trigger an alert.

aligned value

A value that is used to interpret equivalent native values from different data sources. For example, the value Male can be used to align the native values M or Ma.

analysis attribute

A characteristic or trait pertaining to a chart item. Analysis attributes are never displayed on charts.

association chart

A chart that highlights the relationships between entities, rather than a chronology of events, by arranging data in a manner that emphasizes particular associations.

attribute

A piece of information that is associated with a chart item, such as a date of birth or an account number. An attribute is represented by a symbol, or a value, or both, that is displayed with the chart item.

attribute class

A descriptor of the characteristics of an attribute, including the type of its values, how its values are displayed, and the treatment of its values when they are merged or pasted on a chart.

attribute entry

An attribute with a preset value that can be associated with a chart item.

attribute instance

A single use of an attribute on a chart item.

audit

To record information about database or instance activity by applications or individuals.

authority

A measure of how well-connected an entity is, based on its inbound links. Authority is one of two eigenvector centrality measures used in social network analysis. See also centrality, eigenvector.

automatic attribute

An attribute that is created automatically by the application and added to a chart item.

B

betweenness

A measure of how important an entity is, based on the number of paths that pass through it on an association chart. Betweenness is one of the centrality measures used in social network analysis. See also centrality, gatekeeper.

binding strength

A measure of the strength of a relationship between two entities that are directly or indirectly linked. See also common neighbor.

box

An entity representation that can indicate an organization or group on a chart. A box is often used to enclose other entities. See also circle, representation.

C

card

A record of information attached to an item. An item can have multiple cards.

centrality

The relative importance of one entity compared to other entities in social network analysis, as determined by its relationships. See also authority, betweenness, closeness, degree, eigenvector, hub, social network analysis.

chart

A visual representation of real-world objects, such as organizations, people, events, or locations, and the relationships between them.

chart fragment

A view of a chart that highlights particular items of interest.

charting scheme

A definition that describes how item data behaves when it is visualized on a chart. For example, how data is copied into chart item properties, the chart template and labeling scheme to use, and whether to display attributes and pictures. See also chart template.

chart property

A characteristic of a chart, such as its summary description, time zone, grid size, background color, or merge and paste rules. Chart properties are saved with the chart. See also chart template.

chart template

An object that is used for chart creation that contains preconfigured chart properties, and lists of permitted entity types and link types. See also chart property, charting scheme.

child

In a generalization relationship, the specialization of another element, the parent. See also parent.

circle

An entity representation that can indicate an organization or a group on a chart. A circle is often used to enclose other entities. See also box, representation.

circular layout

A layout in which entities are arranged by type around the circumference of a circle. See also layout.

cloaked item

An item whose existence is known to the user but whose information is hidden from the user. See also item, placeholder, signpost message.

closeness

A measure of how quickly an entity can use links to get access to other entities on an association chart. Closeness is one of the centrality measures used in social network analysis. See also centrality.

cluster

A group of entities that have more connections to each other than to entities outside the group.

common neighbor

An entity that is directly connected to at least two other entities. For example, if C is connected to A and B, then C is a common neighbor of A and B. See also binding strength, connection.

compact peacock layout

A layout in which complex groups of linked entities are arranged to highlight the structure of associations. It is most suitable for charts with many linked entities. See also layout.

condition

A specified property, a value, and an operator that defines a comparison relationship between them. One or more conditions can be used to create a query or a conditional formatting specification. See also parameterized query.

conditional formatting

The process of defining and applying rules to change the appearance of chart items automatically, based on their properties. See also conditional formatting specification.

conditional formatting specification

A collection of conditional formatting rules. See also conditional formatting.

connection

A direct relationship between a pair of entities on a chart, represented by one or more links. See also common neighbor, connection multiplicity, directed connection.

connection multiplicity

A setting that controls whether multiple links between the same items are displayed as a single line, as directed lines, or as multiple lines. See also connection.

controlling item

A chart item whose position on the chart is defined by its date and time, and whose position affects the positions of other timed items. See also free item, ordered item.

cover sheet

A page on which the user can view and edit the summary and custom properties of a chart.

D

degree

A measure of how many direct relationships an entity has with other entities on an association chart. Degree is one of the centrality measures used in social network analysis. See also centrality, root entity.

directed connection

A connection between entities in which links that are in the same direction are represented as a single link on a chart. See also connection.

diverted theme line

A theme line that is attached to an event frame such that when the event frame is moved, the theme line maintains its vertical position with respect to the frame. See also event frame, theme line.

E

eigenvector

A measure of how well-connected an entity is, based on its inbound and outbound links. Eigenvector is one of the centrality measures used in social network analysis. See also authority, centrality, hub.

end

An entity that is attached to a link. See also end constraint.

end constraint

A constraint on the types of entities that can be the end of a particular link. See also end, valid end type.

entity

A set of details that are held about a real-world object such as a person, location, or bank account. An entity is a kind of item.

entity semantic type

A semantic type that can be assigned only to an entity or an entity type. See also semantic type.

entity type

A descriptor of the characteristics of an entity, including the properties it can contain and its appearance in visualizations.

event frame

An entity representation that emphasizes date and time information. An event frame is often used in conjunction with theme lines. See also diverted theme line, representation.

excluded word list

A list of words that are ignored when they are entered as search terms.

expansion

A process that searches for entities within a data source that are directly related to some selected entities.

F

free item

A chart item that is not ordered. Free items can be moved anywhere on the chart. See also controlling item, ordered item.

G

gatekeeper

An entity with a high measure of betweenness that may control the flow of information among other entities on an association chart. See also betweenness.

grade

A rating that indicates the accuracy of a piece of information or the reliability of an intelligence source.

grading system

A rating scale that is used to classify information in a data store or on a chart. A grading system is a measure of reliability and accuracy.

grant level

A measure of the rights that a user has to change the security permissions of an item. Grant levels are calculated separately for every user and every item. See also access level, security dimension.

grouped layout

A layout in which entities are arranged to show groups of interconnected entities. See also layout.

H

heat map

A graphical representation of data values in a two-dimensional table format, in which higher values are represented by darker colors and lower values by lighter ones.

hierarchical layout

A layout in which entities are arranged to show organizational structures. See also layout.

histogram

A graphical display of the distribution of values for a numeric field, in the form of a vertical bar chart in which taller bars indicate higher values. See also histogram filter.

histogram filter

A filter that changes the appearance of a chart. When a histogram bar is selected, items that match the conditions defined by that bar are selected, while items that do not are hidden. See also histogram.

hub

A measure of how well-connected an entity is, based on its outbound links. Hub is one of two eigenvector centrality measures used in social network analysis. See also centrality, eigenvector.

I

icon

An entity representation that consists of a stylized image and an optional label. See also representation.

import design

A specification of how data from an external source will be transformed into chart items during an import procedure.

item

An entity or a link. Items are characterized by the values of their properties. See also cloaked item, merged item, ordered item.

L

labeling scheme

A specification for combining property values to be displayed on screen, or as chart item labels.

layout

The arrangement of items on a chart. See also circular layout, compact peacock layout, grouped layout, hierarchical layout, minimize crossed links layout, peacock layout.

line strength

An indication of confidence in the information underlying a particular link. Line strength is represented as a solid, dashed, or dotted line on a chart.

link

An association between two entities, such as an ownership relationship between a person and a vehicle.

link direction

An indication that the meaning of a link is different for each of its ends. For example, the direction of a telephone call makes one end the caller and the other the recipient. Link direction can influence the centrality measures used in social network analysis.

link semantic type

A semantic type that can be assigned only to a link or a link type. See also semantic type.

link separation

The distance between adjacent links in a connection on a chart.

link type

A descriptor of the characteristics of a link, including the properties it can contain and its appearance in visualizations.

M

match

The part of a result that met a condition during a search operation. A search can yield a perfect match or a partial match.

merged item

An item that is created by merging the information held in two or more items. See also item.

minimize crossed links layout

A layout in which entities are arranged in a configuration where the fewest number of links overlap. See also layout.

multiplicity

See connection multiplicity.

N

network chart

See association chart.

O

ordered item

A chart item whose position is maintained within a sequence. The movement of an ordered item is restricted such that it cannot be dragged beyond neighboring ordered items. See also controlling item, free item, item.

P

parameterized query

A query with conditions in which one or more parameters are defined. The parameter values are set by the user. See also condition.

parent

In a hierarchy or auto-level hierarchy, a member that has one or more child members at the level immediately below.

path

A route on a chart between two entities. A path may include intermediate entities.

peacock layout

A layout where complex groups of linked entities are arranged to show the structure of associations. It is most suitable for charts with many linked entities. See also layout.

pick list

A data category that has a limited number of permissible values, which are often presented in a drop-down list in the user interface.

placeholder

A redacted version of an item that is displayed to the user in situations where displaying the full item is not possible or not permitted. See also cloaked item, signpost message.

property

A container for a single piece of information about an item.

property group

A piece of information about an item that comprises related properties. For example, a distinguishing feature of a person comprises information about the type, appearance, and location of the distinguishing feature.

property semantic type

A semantic type that can be assigned to a property type, a property in a data record, or an attribute class. See also semantic type.

property type

A descriptor of the characteristics of a property, including the type of information it can contain.

proportional

Pertaining to an area of a chart in which the horizontal distances between items have a linear relationship with the time differences between them.

R

representation

The form in which an entity is represented on a chart. See also box, circle, event frame, icon, theme line.

root entity

An entity in a grouped layout that has the highest degree centrality in its group. Depending on the data, there can be more than one root entity. See also degree.

S

schema

A complete description of all the entity types, link types, and their associated property types that are available for items within a system.

security dimension

A collection of related values that can be used to label a user according to their role or security clearance, with the aim of affecting their access to information. See also access level, grant level.

semantic type

A category that defines the real-world meaning of data, and therefore how applications should interpret that data. For example, Person is a semantic type that could be assigned to entity types such as Male, Victim, and Witness. See also entity semantic type, link semantic type, property semantic type.

signpost message

A piece of text that is stored and displayed with an item or a placeholder. The signpost message explains how to obtain more control over, or more information about, the item. See also cloaked item, placeholder.

snapshot

A stored version of a chart that preserves its contents and layout at a particular stage of its development.

social network analysis

A method of analyzing the structure of social relationships that uses mathematical metrics to make claims about social organization and social dynamics. See also centrality, weight.

source reference

An identifier that indicates the source of information, for example, a document reference number.

style segment

A section of a theme line between adjacent items to which color and strength can be applied.

T

theme line

An entity representation that shows the interactions of an entity over time. A theme line can be used with event frames. See also diverted theme line, representation.

theme line extent

The distance between the beginning and end of a theme line

theme line wiring

The manner in which a theme line diverts from a horizontal trajectory in order to pass through and travel between event frames.

timeline chart

A chart or a portion of a chart that shows a chronology of events. For example, a series of meetings that occur over several days, or a set of transactions that occur over a period of time.

V

valid end type

An entity type that conforms to the end constraints of a particular link. See also end constraint.

W

weight

A value that is added to a link on an association chart, to represent its importance relative to other links. Weight can influence the centrality measures used in social network analysis. See also social network analysis.

weightings file

A file that contains information that can apply weighting values to links on a chart.

wiring segment

The section of a theme line between adjacent diverting event frames.

i2 Analyze security schemas

An i2 Analyze security schema defines the security dimension values that can be assigned to records, and the security permissions that you assign to users. You must develop a security schema that meets the requirements of your organization.

All deployments of i2 Analyze include an XML file that contains definitions of security dimensions and permissions for that deployment. Optionally, the file can also refer to *providers* that add to the schema at runtime by implementing a Java SPI:

- The definitions of security dimensions in the file can refer to *dimension values providers* that can add values to a security dimension dynamically.
- The file can also refer to a *permissions provider* that adds to the permissions a user has whenever they interact with the system.

The security schema that the i2 Analyze server uses is a composite of the definitions in the XML file, and the additions that any specified dimension values providers or permissions providers make.

Note: Records in the Information Store contain security dimension values. Major changes to the security schema - for example, adding or removing dimensions - can invalidate records and require you to clear and re-ingest or repopulate data. Behavior like that is not reasonable after you deploy into production, and so it's important to refine some aspects of the security schema before you do so.

Creating a security schema

Every deployment of i2 Analyze requires a security schema file that encapsulates the security model for that deployment. The easiest way to create the file is to start from the example that i2 provides with the platform.

Before you begin

Try to consider the following aspects of your security model before you create the XML security schema file:

- Decide on the security dimensions for categorizing the records in your deployment.

  Adding, removing, and modifying dimension *values* is relatively easy, but changing the dimensions themselves can be an expensive operation.
- Identify the user groups to which security permissions are to be assigned.

  When you deploy i2 Analyze, the group names in your security schema must match the names of system user groups in your user registry.

About this task

An i2 Analyze security schema file contains definitions of security dimensions and security permissions. When you create the file, you define the dimensions first, and then define the security permissions that refer to them.

Note: Optionally, you can specify that values of a particular security dimension, or the permissions for a particular user, are provided or added while the server is running. For more information, see Providing security dimension values and Providing security permissions. The procedure here assumes that everything is defined in the XML file.

Procedure

The following steps describe how to create and configure a security schema file.

1. Navigate to the directory in the deployment toolkit that contains the example security schema: `toolkit\configuration\examples\security-schema`.
2. Make a copy of the `example-dynamic-security-schema.xml` file, give it an appropriate name, and then open it in an XML editor.
3. Edit the contents of the `<AccessSecurityDimensions>` element so that it contains a `<Dimension>` element for each category that your deployment uses to determine access rights to i2 Analyze records.
4. Edit the contents of the `<SecurityPermissions>` element:
   a. Add or modify `<GroupPermissions>` elements so that they reflect all the user groups to which you assign security permissions. The group names in your security schema must match the names of system user groups.
   b. Within each `<GroupPermissions>` element, add or modify `<Permissions>` elements to indicate which dimensions are affected by membership of each user group.

c. Within each <Permissions> element, add or modify <Permission> elements to assign access levels to records that have particular dimension values.

   There are two permitted values for the Level attribute of the <Permission> element:

   - READ_ONLY
   - UPDATE

   For each <Permissions> element, any dimension value that does not feature in a <Permission> element is implicitly associated with the NONE access level.

5. Edit the contents of the <DefaultSecurityDimensionValues> element to define the default security dimension values that i2 Analyze provides to any records that users create.

   Update the value of the <DefaultSecurityDimensionValues> element with a comma-separated list of identifiers of security dimension values that you want to be applied by default. For example, <DefaultSecurityDimensionValues>CON,OSI,HI</DefaultSecurityDimensionValues>.

6. Save the completed security schema file to the configuration\fragments\common\WEB-INF\classes directory in the deployment toolkit.

Configuring the security schema

After you create and deploy an i2 Analyze security schema file, there are restrictions on the changes that you can make to it. The actions that you perform to update a deployed security schema file depend on the changes that you make.

If you want to *modify* a security schema, you are relatively free to:

- Change any display names
- Add security dimension values to existing security dimensions
- Remove dimension values from existing dimensions
- Create security groups and permissions

For more information about performing these types of change, see Modifying security dimensions and Modifying security permissions.

If you want to *replace* the security schema, make other destructive changes, or add security dimensions, you must remove and re-create the underlying database in the system. For more information about performing these types of change, see Replacing a security schema.

Modifying security dimensions

Depending on how your deployment of i2 Analyze is configured and what your aims are, modifying the security dimensions in the security schema can require different approaches. The implications of any changes, for you and your users, can also be different.

Permitted changes to security dimensions

The following table shows the changes that you can make to the dimensions in the security schema of a deployed i2 Analyze server. Some changes to ordered dimensions require you to re-create the search index, which can be time-consuming.

Making a change that's not allowed requires clearing the data from the system. It's important to be sure about the security dimensions you need before you go into production!

The table is written in terms of how dimensions and dimension values are defined in the security schema XML file. However, the same rules apply when the values from a dimension values provider change.

Change	**XML elements or attributes**	**Allowed**	**Reindex?**
Add a security dimension	`<Dimension>`	No	N/A
Modify an existing security dimension	`DisplayName`, `Description`, `ResolutionMode`	Yes	No
Remove an existing security dimension	`<Dimension>`	No	N/A
Add a dimension value to an *unordered* security dimension	`<DimensionValue>`	Yes	No
Add a dimension value to an *ordered* security dimension	`<DimensionValue>`	Yes	Yes*
Reorder the dimension values in an ordered security dimension	`<DimensionValue>`	Yes	Yes
Modify an existing dimension value	`DisplayName`, `Description`	Yes	No
Remove an existing dimension value from a security dimension	`<DimensionValue>`	Yes	No**
Restore a previous dimension value to a security dimension***	`<DimensionValue>`	Yes	No
Move an existing dimension value between security dimensions	`<DimensionValue>`	No	N/A

* If you add a dimension value to the bottom of the sequence in an ordered security dimension, you do not need to complete a reindex.

** Removing a dimension value does not require a reindex, but you should update any ingested data so that it does not use that value. See What to do next after editing the security schema file.

*** If you restore a dimension value to an ordered security dimension, you must also restore the previous order to avoid a reindex.

Editing dimensions in the security schema file

If you're not using security dimension values providers, modifying the security dimensions of a deployed i2 Analyze server means editing the security schema file. After you change the security dimensions, you might also need to update the data in your system.

About this task

To modify the display name or the description of a dimension or a dimension value, you change the `DisplayName` or `Description` attributes of an existing `<Dimension>` or `<DimensionValue>` element. Do not change the value of the `Id` attribute.

To add a security dimension value to a security dimension, you add a `<DimensionValue>` element as a child of an existing `<Dimension>` element.

To remove a security dimension value from a security dimension, you remove the corresponding `<DimensionValue>` element.

Note: If the deployment contains records with dimension values that you remove, users of i2 Analyze client software see those values marked as *suspended* in the user interface. They don't contribute to access level calculations and cannot be assigned to records unless you restore them.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

Edit the security schema:

1. Using an XML editor, open the security schema file for the deployment.

   The security schema file is in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory. The name of the file is specified in the `DynamicSecuritySchemaResource` property of the `ApolloServerSettingsMandatory.properties` file in the same directory.
2. Modify the security dimensions in the security schema file according to your requirements.
3. Remove or edit any security permissions that refer to dimension values that are no longer in the schema. Add security permissions for any dimension values that you added.
4. Check the updated schema to ensure that it remains possible for all users to get the "Read only" or "Update" access level for at least one value in every dimension.
5. Increment the version number that is stated in the `Version` attribute of the `<SecurityDimensions>` element in the security schema file.
6. Save and close the file.

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. If you completed a change that requires a reindex, clear the search index:

   ```
   setup -t clearSearchIndex --hostname <liberty.hostname>
   ```

In a deployment that provides high availability, you only need to run this command on one Liberty server.

4. Update and redeploy the system:

```
setup -t updateSecuritySchema
setup -t deployLiberty
```

5. Start Liberty:

```
setup -t startLiberty
```

What to do next

If your changes to the security schema included removing dimension values, records with those values remain in the system. The records might be in the Information Store as a result of ingestion or user upload; or they might be on charts that users have created. The removed values are not harmful, but they are potentially confusing, and you should try to delete them from existing data.

For system-governed records that you add to the Information Store through an ETL pipeline, you can modify the ingestion mappings so that the removed values are no longer used, and then ingest the records again. Records on users' charts are updated automatically when they connect to the server.

For analyst-governed records that users create, users who inspect the security settings of a record see **(suspended)** after the name of any dimension value that you removed. If they have "Update" access to the record, they can edit its security settings to deselect suspended values and select new ones as appropriate.

Whenever you make changes to the security schema, keep your users' experience in mind, and inform them of any changes they'll see (or need to make) as a result.

Providing security dimension values

If an unordered dimension in your i2 Analyze security schema has values that can change often, or if you have an existing classification system that you want to adopt without duplicating it in the schema, you can use a *security dimension values provider* to supply some or all of the values for a security dimension.

Before you begin

To use a security dimension values provider in a security schema file, you must create or acquire the Java class that contains it. For more information about creating a provider, see i2 Analyze Developer Essentials.

About this task

To specify that a dimension in your security schema uses values from a security dimension values provider, you add the name of the class to a `<Dimension>` element in the security schema file, and make the class available to the i2 Analyze application by editing the topology file.

i2 Analyze places no restrictions on the number of security dimension values providers you can use. Each provider *adds* values to its associated dimension. If a dimension in the file contains no dimension values, then the provider supplies all the values for that dimension.

Note: The security dimension itself must always be defined in the security schema file. It is not possible to add or remove security dimensions dynamically.

Procedure

Configure a security dimension to use a dimension values provider:

1. Using an XML editor, open the security schema file for the deployment.

   The security schema file is in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory. The name of the file is specified in the `DynamicSecuritySchemaResource` property of the `ApolloServerSettingsMandatory.properties` file in the same directory.
2. In the `<Dimension>` element for the dimension that the provider works with, add the `ProviderClass` attribute. For example:

   ```
   <Dimension Id="SD-SC"
       DisplayName="Security Compartment"
       Description="Security Compartment"
       Ordered="false"

    ProviderClass="com.example.security.provider.SecurityCompartmentDimensionValuesProvi
   >
   ```
3. Modify the security permissions so that it's possible for all users to get the "Read only" or "Update" access level for at least one value in the updated dimension.

   Note: When you use a dimension values provider, it's likely that you'll also need to use a security permissions provider.
4. Increment the version number that is stated in the `Version` attribute of the `<SecurityDimensions>` element in the security schema file.
5. Save and close the file.

Update the topology file so that i2 Analyze can retrieve values from the provider:

1. Copy the JAR file that contains the security dimension values provider, as well as any dependencies of that file, to a new subdirectory of the `fragments` directory. For example, `toolkit\configuration\fragments\security-compartment-provider`.
2. Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
3. Edit the `<fragments>` element to include the new fragment. For example:

   ```
   <fragments>
     <fragment name="opal-services-is"/>
     <fragment name="opal-services"/>
     <fragment name="common"/>
     <fragment name="default-user-profile-provider"/>
     <fragment name="security-compartment-provider"/>
   </fragments>
   ```

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. Update and redeploy the system:

   ```
   setup -t updateSecuritySchema
   setup -t deployLiberty
   ```
4. Start Liberty:

   ```
   setup -t startLiberty
   ```

After you've configured the i2 Analyze server to use a dimension values provider, you can change the values that it provides without reconfiguring the server.

The server requests values from all providers at startup, and at intervals afterwards, to ensure that it always has an up-to-date set.

What to do next

Security dimension values must have security permissions associated with them. The new provider adds security dimension values to the security schema dynamically, and it is likely that you'll need to deploy a security permissions provider to add those dynamically too.

If you use a security dimension values provider that might stop providing some values, the effect is the same as removing values from the security schema file. It has the same potential impact on users, and you should take the same steps to explain the behavior they see.

Reloading dimension values from providers

By default, i2 Analyze reloads security dimension values from providers every 15 minutes. If your deployment requires it, you can change that period to be shorter or longer, or instruct the server to reload them immediately.

Changing the security dimension values reload interval

To change the interval between requests from i2 Analyze to security dimension values providers, you edit a setting in the `DiscoServerSettingsCommon.properties` file:

1. Using a text editor, open the file at `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties`.
2. Edit the following settings, and then save and close the file:

   ```
   SecurityDimensionValuesReloadInterval=15
   SecurityDimensionValuesReloadIntervalUnits=MINUTES
   ```
3. In a command prompt, navigate to the `toolkit\scripts` directory.
4. Stop Liberty, update the i2 Analyze application, and restart Liberty:

   ```
   setup -t stopLiberty
   setup -t deployLiberty
   setup -t startLiberty
   ```

Reloading security dimension values from all providers

To make the server reload dimension values, you use an endpoint in the i2 Analyze REST API.

1. Follow the instructions in Using the admin endpoints to set up `curl` and authenticate with the i2 Analyze server.
2. Assuming that your authentication cookie is in `cookie.txt`, execute the following command to reload the security schema:

   ```
   curl -i --cookie cookie.txt
        -X POST
        http://<host_name>/<context_root>/api/v1/admin/securityschema/reload
   ```

Modifying security permissions

It is possible to change the mapping between user groups and the security permissions that the security schema defines without reimporting or reindexing your data. You must take care to ensure that all i2 Analyze users retain the ability to access your deployment.

Permitted changes to security permissions

The following table shows the changes that you can make to the security permissions in a deployed security schema without clearing data from the system:

Change	XML elements or attributes	Allowed	Reindex?
Add a security group	`<GroupPermissions>`	Yes	No
Modify an existing security group	`UserGroup`	Yes	No
Remove an existing security group	`<GroupPermissions>`	Yes	No
Add a security dimension to a security group	`<Permissions>`	Yes	No
Remove a security dimension from a security group	`<Permissions>`	Yes	No
Add security permissions to a security group	`<Permission>`	Yes	No
Modify the access level that a security permission specifies	`DimensionValue`, `Level`	Yes	No
Remove a security permission from a security group	`<Permission>`	Yes	No

Editing permissions in the security schema file

If you're not using a security permissions provider, modifying the security permissions of a deployed i2 Analyze server means editing the security schema file.

About this task

If the requirements for security groups change, you can modify the `<GroupPermissions>` element and its children:

- To add a group, insert a complete <GroupPermissions> element. To use the new group, you must ensure that the user registry contains a group that matches the value of the UserGroup attribute.
- To modify the name that is associated with a group, change the value of the UserGroup attribute.
- To remove a group, remove the <GroupPermissions> element for that group.

If the requirements for the permissions of a security group change, you can add or remove <Permissions> elements, and add, modify, and remove child <Permission> elements.

- To change which dimensions are affected by membership of a particular group, you can add or remove <Permissions> elements as follows:
  - To add to the dimensions whose values appear in the permissions that group members receive, insert a <Permissions> element whose Dimension attribute matches the value of the Id attribute of the dimension.
  - To remove a dimension so that its values no longer appear in the permissions that group members receive, delete the <Permissions> element whose Dimension attribute matches the value of the Id attribute of the dimension.
- To change the security permissions that group members receive for the values of a particular dimension, you can add, modify, and remove <Permission> elements as follows:
  - To add a permission to a group, insert a <Permission> element. The DimensionValue attribute must match a value in the dimension that's identified in the Dimension attribute of the parent <Permissions> element.
  - To modify the current permission that a group has for a dimension value, set the Level attribute to a different value.
  - To modify the dimension value that a permission is for, set the DimensionValue attribute to a different value.
  - To remove a permission that a group has for a dimension value, remove the <Permission> element in which the DimensionValue attribute matches that dimension value.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

Edit the security schema:

1. Using an XML editor, open the security schema for the deployment.

   The security schema is in the toolkit\configuration\fragments\common\WEB-INF\classes directory. The name of the security schema is specified in the DynamicSecuritySchemaResource property of the ApolloServerSettingsMandatory.properties file in the same directory.
2. Modify the security permissions in the security schema according to your requirements.
3. Increment the version number that is stated in the Version attribute of the <SecurityDimensions> element in the security schema.
4. Check your updated schema to ensure that it remains possible for all users to get a "Read only" or "Update" access level for at least one value in every dimension.
5. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. Update and redeploy the system:

   ```
   setup -t updateSecuritySchema
   setup -t deployLiberty
   ```
4. Start Liberty:

   ```
   setup -t startLiberty
   ```

Providing security permissions

In an i2 Analyze security schema, you can use a *security permissions provider* to return permissions that add to those in the security schema file. i2 Analyze asks the provider for permissions every time a user makes a request that requires a security calculation.

Before you begin

There are two reasons for providing security permissions dynamically that can both be true at the same time:

- If your security schema uses dimension values providers, the only way to give users permissions that use those provided values is through a security permissions provider.
- A security permissions provider is the only way to allocate permissions on a strictly per-user, rather than a per-group, basis. i2 Analyze sends the name *and* system group memberships of the current user to the provider when it asks for permissions.

To use a security permissions provider in a security schema file, you must create or acquire the Java class that contains it. For more information about creating a provider, see i2 Analyze Developer Essentials.

About this task

A security schema can use one permissions provider. To make it do so, you add the name of the class to the `<SecurityPermissions>` element in the security schema file, and make the class available to the i2 Analyze application by editing the topology file.

1. Using an XML editor, open the security schema file for the deployment.

   The security schema file is in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory. The name of the file is specified in the `DynamicSecuritySchemaResource` property of the `ApolloServerSettingsMandatory.properties` file in the same directory.
2. In the `<SecurityPermissions>` element, add the `ProviderClass` attribute. For example:

   ```
   <SecurityPermissions
    ProviderClass="com.example.security.provider.SecurityPermissionsProvider">
       ...
   </SecurityPermissions>
   ```
3. Increment the version number that is stated in the `Version` attribute of the `<SecurityDimensions>` element in the security schema file.
4. Save and close the file.

Update the topology file so that i2 Analyze can use the provider:

1. Copy the JAR file that contains the security permissions provider, as well as any dependencies of that file, to a new subdirectory of the `fragments` directory. For example, `toolkit\configuration\fragments\security-permissions-provider`.
2. Using an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
3. Edit the `<fragments>` element to include the new fragment. For example:

```
<fragments>
  <fragment name="opal-services-is"/>
  <fragment name="opal-services"/>
  <fragment name="common"/>
  <fragment name="default-user-profile-provider"/>
  <fragment name="security-permissions-provider"/>
</fragments>
```

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

```
setup -t stopLiberty
```

3. Update and redeploy the system:

```
setup -t updateSecuritySchema
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

Setting default dimension values

When users create i2 Analyze records in Analyst's Notebook, i2 Analyze applies a default set of dimension values to each record. You specify those dimension values in the security schema file.

About this task

You might change the default security dimension values that are applied to records for different reasons:

- If you change the dimensions or dimension values in your security schema
- If the security requirements of your deployment change.

Regardless of your reason for changing the default security dimension values, or when the values are applied, each record must have at least one dimension value from each security dimension.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using an XML editor, open the security schema file for the deployment.

   The security schema file is in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory. The name of the file is specified in the `DynamicSecuritySchemaResource` property of the `ApolloServerSettingsMandatory.properties` file in the same directory.
2. Update the value of the `<DefaultSecurityDimensionValues>` element with a comma-separated list of identifiers of security dimension values that you want to be applied by default. For example, `<DefaultSecurityDimensionValues>CON,OSI,HI</DefaultSecurityDimensionValues>`

In a standard deployment of i2 Analyze, a supplied implementation of the `DefaultSecurityDimensionValuesProvider` that applies dimension values from the `<DefaultSecurityDimensionValues>` element is used. You can create your own implementation that does not use these values by using i2 Analyze Developer Essentials.

3. In a command prompt, navigate to the `toolkit\scripts` directory.
4. Stop Liberty:

```
setup -t stopLiberty
```

5. Update and redeploy the system:

```
setup -t updateSecuritySchema
setup -t deployLiberty
```

6. Start Liberty:

```
setup -t startLiberty
```

Note: The default dimension values that user-created records receive have no connection with the dimension values that ingested records receive. For information about those values, see Ingestion mapping files.

Replacing a security schema

The security schema is a key component of an i2 Analyze deployment, and configuring it correctly is an important part of the development process. Replacing or making some destructive changes to the security schema requires you to clear and repopulate the data stores in your deployment.

About this task

In a deployment that provides high availability, use the documentation from your database management system to remove the Information Store database from each database server in your deployment instead of using the `dropDatabases` toolkit task.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `deleteSolrCollections` and `createSolrCollections` on one Liberty server only.

Procedure

Edit the security schema:

1. Modify or create the security schema that you want to update your deployment with. For more information about creating the security schema, see Creating a security schema.
2. Update the configuration with your security schema.
   a. Ensure that the security schema file is in the `configuration\fragments\common\WEB-INF\classes` directory.
   b. Ensure that your security schema file is specified in `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties`.
   c. Ensure that the identifiers of the security dimension values that records receive by default are valid in the `<DefaultSecurityDimensionValues>` element in your security schema. For more information, see Setting default dimension values.

The following steps update your deployment with the new security schema:

1. Stop the deployment:

```
setup -t stopLiberty
```

2. To remove the database and Solr collections, navigate to the `toolkit\scripts` directory and run the following commands:

   ```
   setup -t dropDatabases --hostname <liberty.host-name>
   setup -t deleteSolrCollections --all --hostname <liberty.host-name>
   ```

   Here, `<liberty.hostname>` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

   A message is displayed when you run each task to confirm that you want to complete the action. Enter `Y` to continue. The database and Solr collections are removed from the system.

3. To re-create the Solr collections and databases, run the following commands:

   ```
   setup -t createSolrCollections --hostname <liberty.host-name>
   setup -t createDatabases
   ```

4. Update and redeploy the system:

   ```
   setup -t updateSecuritySchema
   setup -t deployLiberty
   ```

5. Start Liberty:

   ```
   setup -t startLiberty
   ```

6. If you changed the names of the user groups in the security schema, update the basic user registry to match the new names. For more information, see Configuring the Liberty user registry.

What to do next

Add some data to the system, and verify that users see the behavior that you intended. Iterate over the process of modifying and replacing the schema as many times as you need.

The security schema file

An i2 Analyze security schema file is an XML file with a relatively simple structure. Security dimensions and security permissions are defined in separate sections of the file.

In outline, the `<SecuritySchema>` root element of a security schema contains child elements for the dimension and permission definitions:

```
<SecuritySchema>
  <SecurityDimensions Id="" Version="">
    <AccessSecurityDimensions>
      <Dimension ...>
        <DimensionValue ... />
        ...
      </Dimension>
      ...
    </AccessSecurityDimensions>
  </SecurityDimensions>

  <SecurityPermissions>
    <GroupPermissions ...>
      <Permissions ...>
        <Permission ... />
        ...
      </Permissions>
      ...
    </GroupPermissions>
    ...
```

```
  </SecurityPermissions>
</SecuritySchema>
```

The <SecurityDimensions> element has attributes for the Id and Version of the schema. If you modify any part of the security schema, retain the identifier but increment the version number. In this way, you ensure that all i2 data stores and services are informed of the changes.

In a valid security schema, the <AccessSecurityDimensions> element must be present, and there must be at least one <GroupPermissions> element inside <SecurityPermissions>.

Security dimension definitions

Security dimensions are defined in an i2 Analyze security schema file as children of the mandatory <AccessSecurityDimensions> element. A valid security schema defines at least one security dimension.

The following example shows a simple, complete <AccessSecurityDimensions> element:

```
<AccessSecurityDimensions>
  <Dimension Id="SD-OT"
             DisplayName="Operational Team"
             Description="The teams whose members have access to this
 information"
             Ordered="false"
             ResolutionMode="ANY">
    <DimensionValue Id="SD-OT-A" DisplayName="Team A" Description="Team A" /
>
    <DimensionValue Id="SD-OT-B" DisplayName="Team B" Description="Team B" /
>
  </Dimension>
</AccessSecurityDimensions>
```

The attributes of the <Dimension> element affect how the values in the security dimension are interpreted.

Attribute	**Description**
Id	A unique identifier that is used to distinguish this security dimension throughout the system.
DisplayName	A name that identifies this dimension to the user in clients.
Description	A more detailed description of this security dimension that provides more information to the user. In Analyst's Notebook, the description is used as a tooltip.
Ordered	Indicates whether the values in this dimension form a descending sequence in which each value supersedes the values below it. The permitted values are true and false.
ResolutionMode	Determines how i2 Analyze calculates access when a record has more than one value from this

Attribute	Description
	dimension. The permitted values are `ANY` and `ALL`.

Note: `ResolutionMode="ALL"` is valid only when `Ordered="false"`. A record cannot have multiple values from a dimension in which `Ordered="true"`. For more information, see i2 Analyze security dimensions.

The `Id`, `DisplayName`, and `Description` attributes of `<DimensionValue>` elements have the same purpose and meaning as the `<Dimension>` attributes with the same names. The identifiers of dimension values must also be unique throughout the security schema.

Important: After you deploy i2 Analyze, the changes that you can make to security dimensions are limited. You cannot add or remove dimensions, or move dimension values between dimensions. For this reason, you must understand the requirements of your organization before you deploy i2 Analyze in a production environment.

Security group permission definitions

In an i2 Analyze security schema, the mandatory `<SecurityPermissions>` element contains one or more `<GroupPermissions>` elements. Each `<GroupPermissions>` element defines the access levels that users in a particular group receive for records with particular dimension values in an i2 Analyze deployment.

The syntax for defining the security permissions for system user groups enables membership of one group to convey permissions across several dimensions, and allows different groups to convey different permissions for the same dimensions. The following example shows how to structure `<GroupPermissions>` elements inside the `<SecurityPermissions>` element:

```
<SecurityPermissions>
  <GroupPermissions UserGroup="Clerk">
    <Permissions Dimension="SD-SC">
      <Permission ... />
    </Permissions>
    ...
  </GroupPermissions>
  <GroupPermissions UserGroup="Manager">
    <Permissions Dimension="SD-SC">
      <Permission ... />
      ...
    </Permissions>
    <Permissions Dimension="SD-IT">
      <Permission ... />
      ...
    </Permissions>
    ...
  </GroupPermissions>
  <GroupPermissions UserGroup="Security Controller">
    <Permissions Dimension="SD-GA">
      <Permission ... />
    </Permissions>
  </GroupPermissions>
</SecurityPermissions>
```

The value of the `UserGroup` attribute of each `<GroupPermissions>` element must match the name of a system group of i2 Analyze users.

The value of the `Dimension` attribute of each `<Permissions>` element must match the identifier of one of the dimensions that is defined in the first part of the schema.

It is normal for `<Permissions>` elements with the same dimension to appear in more than one `<GroupPermissions>` element:

- Users who are members of one group but not the other can receive different access levels on records that have the same dimension values.
- When users are members of more than one group, `<Permissions>` elements for the same dimension are combined before any access level calculation takes place.

Important: You can add and remove `<GroupPermissions>` elements from a deployed security schema, if the resulting system continues to obey the rules of i2 Analyze. In particular, it must remain possible for all users to get a "Read only" or "Update" access level for at least one value in every dimension.

Security permission definitions

The security permission definitions in an i2 Analyze security schema each associate a single dimension value with a single access level. The definitions can be simple because of the additional context that their location in the security schema file provides.

The `<Permission>` elements that define security permissions always appear inside `<Permissions>` elements, which in turn always appear inside `<GroupPermissions>` elements.

```
<GroupPermissions UserGroup="Manager">
  <Permissions Dimension="SD-SC">
    <Permission DimensionValue="TOP" Level="UPDATE" />
    <Permission DimensionValue="RES" Level="UPDATE" />
  </Permissions>
  <Permissions Dimension="SD-IT">
    <Permission DimensionValue="HUMINT" Level="READ_ONLY" />
  </Permissions>
</GroupPermissions>
```

It is possible, and often desirable, for identical `<Permission>` elements to appear in different locations in an i2 Analyze security schema. The effect of a security permission definition depends entirely on its position in the file.

Important: Like the `<GroupPermissions>` elements that contain them, you can add and remove `<Permissions>` and `<Permission>` elements from a deployed security schema, as long as the resulting system does not break the rules of i2 Analyze.

User security

In a deployment of i2 Analyze, configuring how users are authenticated and authorized with the system is key to establishing a secure environment. i2 Analyze supports a range of authentication mechanisms, and can be integrated with external identity providers.

Through the processes of authentication and provisioning, i2 Analyze determines whether a user can access the system at all, and associates them with a unique identifier, a user name, and a display name. Users become members of groups, which also have identifiers and display names.

When a user is authenticated, i2 Analyze uses the security schema to authorize their access to data, and command access control to authorize their use of application features.

User authentication

i2 Analyze supports user authentication through two main mechanisms: registries like Active Directory or the Liberty user registry, and claims-based authentication through an identity provider. Typically, the mechanism you choose will depend on your organization's existing infrastructure and security requirements.

When users log in to i2 Analyze, their group memberships determine what data they're allowed to see and what actions they're allowed to perform. So, if the groups that you already have align with the groups that you want to use in i2 Analyze, then after you configure authentication you can start to refine your security schema and configure access to features.

If the groups in your authentication system *don't* align with what you need in i2 Analyze, then you can either create groups specifically for i2 Analyze, or align i2 Analyze with your existing groups. After that, you can configure provisioning to filter out users and groups that you don't want to use in i2 Analyze, or to make other changes such as providing display names for users.

Configuring claims-based authentication

When i2 Analyze uses an identity provider for authentication, authenticated users are associated with *claims* that provide information about them, including their group memberships. i2 Analyze supports user authentication through Security Assertion Markup Language (SAML) and OpenID Connect (OIDC) identity providers.

Configuring i2 Analyze for claims-based authentication

To use any kind of claims-based authentication with i2 Analyze, you must first configure the system to use the claims-based user groups provider.

1. Claims-based authentication requires that your deployment is configured for TLS. For more information, see Transport Layer Security connections with i2 Analyze.
2. Update the `configuration\fragments\common\WEB-INF\classes\ApolloServerSettingsMandatory.properties` file to specify the claims-based user groups provider in the `UserGroupsProvider` setting:

   ```
   UserGroupsProvider=com.i2group.disco.user.ClaimsBasedUserGroupsProvider
   ```

After you've made those changes, you can set up i2 Analyze to use either a SAML or an OIDC identity provider for user authentication.

Configuring SAML authentication

To configure SAML authentication, you must update the Liberty configuration to work with your identity provider and redeploy the system.

1. Update the `configuration\liberty\server.extensions.xml` file to include the `samlWeb-2.0` feature:

   ```
   <featureManager>
     ...
     <feature>samlWeb-2.0</feature>
   </featureManager>
   ```

2. Redeploy and restart Liberty by running the following commands:

   ```
   setup -t deployLiberty
   setup -t startLiberty
   ```

3. Navigate to `https://<liberty-hostname>:<ssl-port>/ibm/saml20/defaultSP/samlmetadata` to download a server provider (SP) metadata file named `spMetadata.xml`.
4. Send the `spMetadata.xml` file to your identity provider and receive an identity provider (IdP) metadata file in exchange. Ensure that you are added to the correct group. You'll be given a group ID that you can use to configure the server.
5. Rename the IdP file that you received to `idpMetadata.xml`, and place it in the `i2analyze\deploy\wlp\usr\servers\opal-server\resources\security\` directory on the Liberty server.
6. Update the `server.xml` file to include a `<samlWebSso20>` element with a `groupIdentifier` attribute. The value for the attribute comes from the URI for the "Groups" claim type in the `idpMetadata.xml` file. For example:

   ```
   <samlWebSso20 id="defaultSP" groupIdentifier="http://
   schemas.microsoft.com/ws/2008/06/identity/claims/groups"/>
   ```

   For more information about all of the options for configuring SAML in Liberty, see SAML Web Single Sign-On 2.0.
7. Ensure that the group names that i2 Analyze receives from the identity provider align with the groups referenced in your security schema and command access control files.

   To update your security schema, follow the instructions in Configuring the security schema.

   Note: If the group names from the identity provider don't match the group names in your security schema, one solution is to use provisioning to map from one to the other.
8. Redeploy and restart Liberty again by running the following commands:

   ```
   setup -t deployLiberty
   setup -t startLiberty
   ```
9. Connect to the system, and log in as a user that is authenticated by your identity provider.

Configuring OIDC authentication

To configure OIDC authentication, you must update the Liberty configuration to work with your identity provider and redeploy the system.

1. Update the `configuration\liberty\server-extensions.xml` file to include the `openidConnectClient-1.0` feature:

   ```
   <featureManager>
     ...
     <feature>openidConnectClient-1.0</feature>
   </featureManager>
   ```
2. In the same file, add an `<openidConnectClient>` element and populate its attributes according to the values from your identity provider.

   The following values are used in the example configuration below:

   - `id` - an identifier for this OIDC client defined in Liberty. This value is used in the redirect URI of the client in the provider.
   - `clientId` - the identifier of the client for i2 Analyze defined in the identity provider.
   - `clientSecret` - if your identity provider is configured for client authentication, this is the secret used to authenticate the client with the provider.
   - `discoveryEndpointUrl` - the URL for the OpenID endpoint configuration from the identity provider.

- `signatureAlgorithm` - the algorithm used to encrypt the tokens that the identity provider returns.
- `userIdentifier` - the name of the claim in the token that contains the user's username.
- `uniqueUserIdentifier` - the name of the claim in the token that contains a unique identifier for the user.
- `groupIdentifier` - the name of the claim in the token that contains a user's group information.

```
<openidConnectClient
    id="client01"
    clientId="oidc-example"
    clientSecret="PUgXRYXhWmVDUbAwHy4Bjg7LiKV"
    discoveryEndpointUrl="https://keycloak.eia:8443/realms/oidc-
example/.well-known/openid-configuration"
    signatureAlgorithm="RS256"
    userIdentifier="preferred_username"
    uniqueUserIdentifier="sub"
    groupIdentifier="groups"
/>
```

For more information about all of the options for configuring OIDC in Liberty, see OpenID Connect Client 1.0.

3. Ensure that the group names that i2 Analyze receives from the identity provider align with the groups referenced in your security schema and command access control files.
4. Redeploy and restart Liberty by running the following commands:

```
setup -t deployLiberty
setup -t startLiberty
```

5. Connect to the system, and log in as a user that is authenticated by your identity provider.

Configuring SPNEGO single sign-on for i2 Analyze

When you configure i2 Analyze to Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) single sign-on, users can access i2 Analyze without having to enter their credentials again. The users and groups in Active Directory become the users and groups that i2 Analyze recognizes and uses to drive authorization.

Intended audience

This section is intended for readers who are familiar with configuring and managing domain controllers, Microsoft Active Directory, and have an understanding of SPNEGO single sign-on.

There are many different single sign-on technologies. This section defines a SPNEGO single sign-on setup with workstations that are members of the same Microsoft Active Directory domain. i2 Analyze uses the users and groups in Active Directory to determine the authorization of users.

The instructions assume that the following prerequisites are installed and accessible:

- A Microsoft Windows Server running an Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC).
- A Microsoft Windows domain member (client) with a web browser that supports the SPNEGO authentication mechanism.
- A working deployment of i2 Analyze that can be accessed by users in Active Directory.

For information on the prerequisites, see the Before you begin section of Configuring SPNEGO authentication.

Attention: i2 takes reasonable steps to verify the suitability of i2 Analyze for internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. i2 accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

i2 Analyze with SPNEGO single sign-on

In the production deployment process, i2 Analyze is configured to use user names and passwords that are stored in a file-based registry. By configuring the deployment to use SPNEGO single sign-on, a user is logged in through the domain client workstation that they are logged in to.

After a user logs in to a single sign-on environment, they are authenticated with any systems that they have access to. i2 Analyze can be configured to allow authentication through SPNEGO single sign-on, with authorization through Active Directory.

SPNEGO single sign-on enables users to log in to a Microsoft domain controller, and be authenticated within the single sign-on environment. In SPNEGO single sign-on, to change the user that is logged in to i2 Analyze, the user must log out of the workstation, and a new user must log in to the workstation.

SPNEGO single sign-on planning

Planning an implementation of SPNEGO single sign-on with i2 Analyze ensures that the system you create matches the needs of your organization. It is important to understand the organizational requirements and your environment before you begin to plan.

Before you start implementing the solution, ensure that you understand the following aspects of the proposed system:

- What SPNEGO single sign-on is, and the implications of implementing it in your environment. For more information, see the SPNEGO section of the Open Liberty documentation for single sign-on.
- The physical architecture that is required to use SPNEGO single sign-on.
- The value of the `UserGroup` attribute of each `<GroupPermissions>` element to use with your i2 Analyze deployment. For more information, see Security group permission definitions.

i2 Analyze with SPNEGO single sign-on model

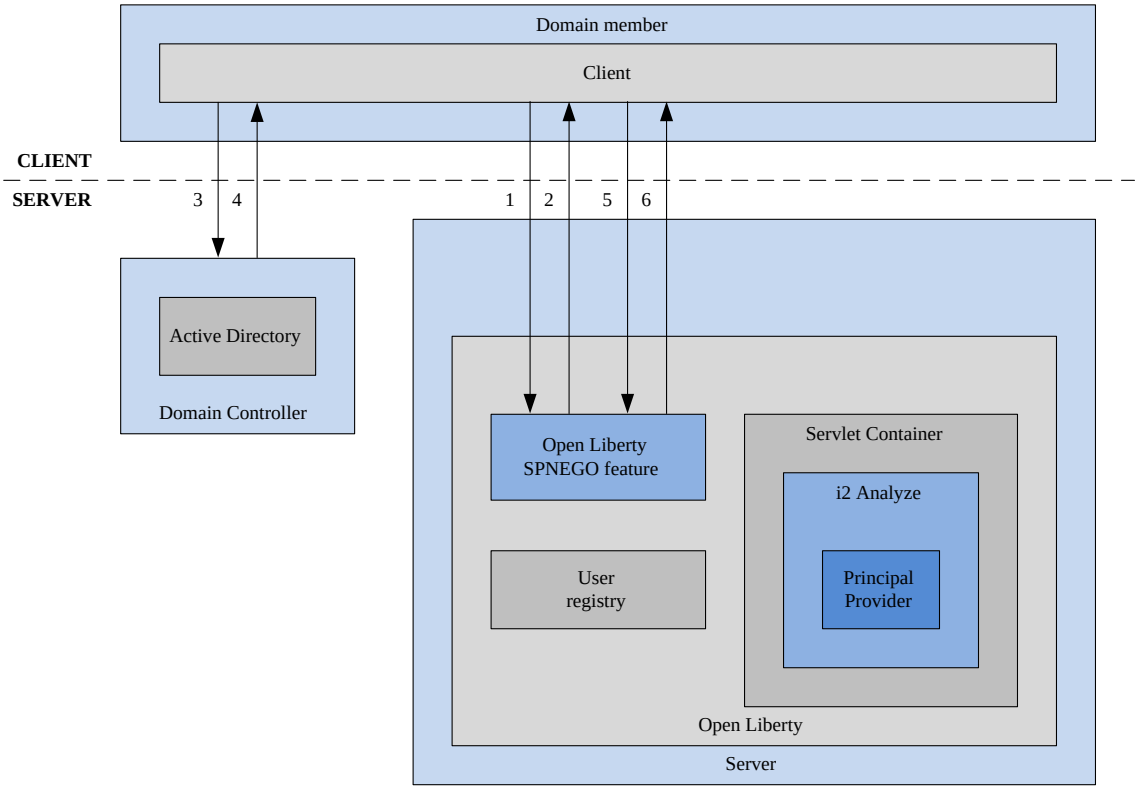
Configuring i2 Analyze to use SPNEGO single sign-on changes the way that users authenticate with the platform. A deployment that uses SPNEGO single sign-on requires the user to access i2 Analyze on a workstation that is a member of the same domain as i2 Analyze.

Authentication

When i2 Analyze is configured to use SPNEGO single sign-on, the authentication sequence between the client and the platform matches the following steps and the associated diagram:

1. The client attempts to connect to the Liberty server with an `HTTP/Post/Get` request.
2. Liberty returns `HTTP 401` with a `Negotiate` header.
3. The client requests a SPNEGO token from the domain controller.
4. The domain controller returns a SPNEGO token to the client.
5. The client attempts to connect to Liberty with an `HTTP/Post/Get` request and the SPNEGO token.

6. On successful authentication, the client receives a Lightweight Third-Party Authentication (LTPA) token in a cookie. During normal operation, the client passes the cookie back to i2 Analyze.

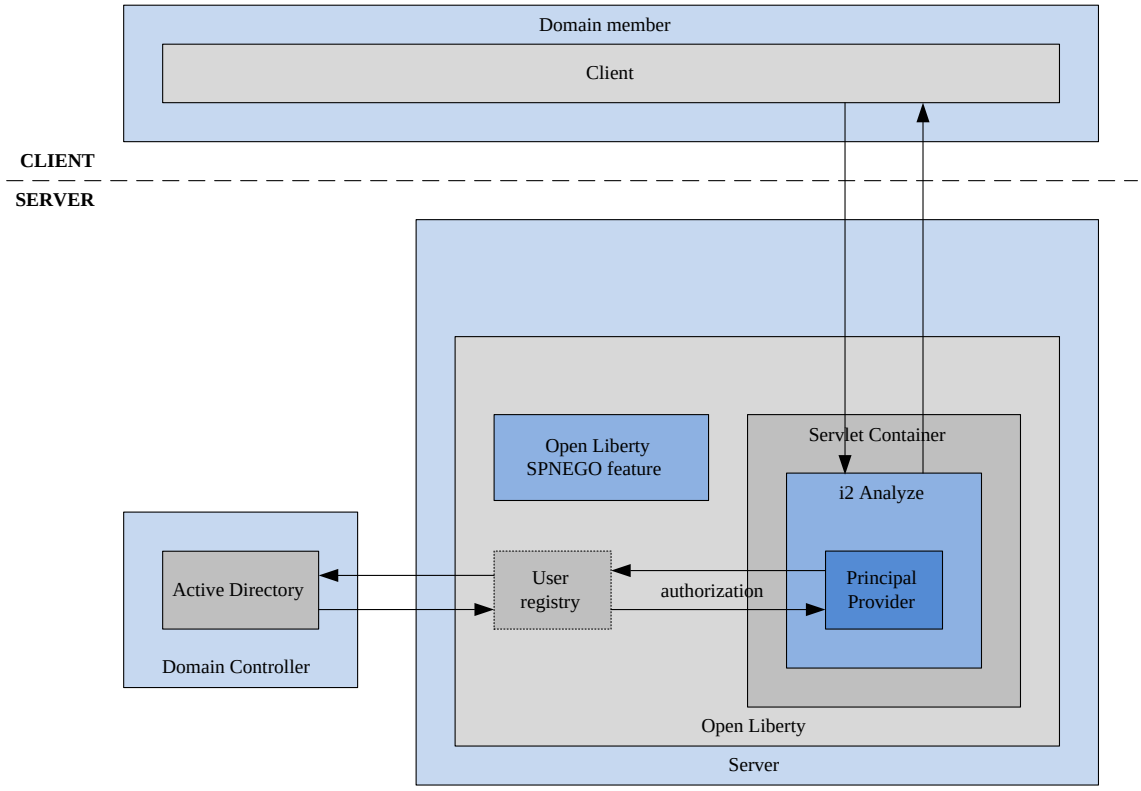


Authorization

After the user is authenticated, they are logged in to i2 Analyze. To define the data that the user has access to, the user must be authorized by i2 Analyze.

For authorization, the i2 Analyze application communicates with Active Directory through the Liberty user registry APIs to retrieve information about the current user. The principal provider then maps the retrieved information to security dimension values in the i2 Analyze security schema.

The following diagram shows how authorization works in i2 Analyze:



Implementing SPNEGO single sign-on

SPNEGO single sign-on enables users to log in to a domain client workstation, and be authenticated with i2 Analyze. Complete any configuration changes to i2 Analyze, and test the system.

Before you begin

For information on the prerequisites, see the Before you begin section of Configuring SPNEGO authentication.

In the production deployment process, you might first configure SPNEGO single sign-on in the configuration or pre-production environments. As you move to a production deployment, you must replicate any configuration changes in any new deployments.

If you previously deployed i2 Analyze with basic authentication, you must remove or comment out the complete `<basicRegistry>` element from your user registry.

About this task

Populate Active Directory with the correct users and system user groups for your environment. Then, complete the configuration steps for Liberty and the i2 Analyze application. Redeploy i2 Analyze, and connect to i2 Analyze from a client workstation.

Configuring Microsoft Active Directory

When you configure i2 Analyze to use SPNEGO, the users that are in Microsoft Active Directory are used to authenticate with i2 Analyze. The groups that are in Active Directory are used to control i2 Analyze functionality including data access, feature availability, and artifact sharing.

About this task

To use membership of Microsoft Active Directory groups as the basis for allowing or denying access to data, *either*:

- the group names must match the values of `UserGroup` attributes of `<GroupPermissions>` elements in the i2 Analyze security schema file, *or*
- you must use provisioning to map Active Directory groups to i2 Analyze groups.

Note: The security schema that the deployment uses is defined in the `ApolloServerSettingsMandatory.properties` file. The security schema and properties files are in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.

In a single sign-on setup, the following users must also be present in Active Directory:

- A user for the server that hosts the i2 Analyze application, which is mapped to a Kerberos Service Principal Name (SPN).
- The users that are used to log in to i2 Analyze.

Procedure

If your organization already uses Active Directory, you can use the existing groups and users to control access to i2 Analyze, and the steps below are not necessary.

Tip: If you want only *some* of the users and groups in Active Directory to be able to access i2 Analyze, you can use provisioning as a filter.

If you're setting up Active Directory for the first time, specifically for use with i2 Analyze, you can create the users and groups that you need.

1. Create the Microsoft Active Directory groups:
   a. Open the Microsoft Active Directory groups controller.
   b. Create groups whose names exactly match the values of `UserGroup` attributes of `<GroupPermissions>` elements in the i2 Analyze security schema file.

   For more information, see How to Create a Group in Active Directory.
2. Create the Microsoft Active Directory user accounts that can be used to log in to i2 Analyze.

   For more information, see How to Create a Domain Account in Active Directory.
3. Make each user a member of the appropriate groups for your environment.

   For more information, see Adding Users to an Active Directory Group.

Results

The users that can access i2 Analyze are created, and are members of the system user groups that govern their access to functionality.

Configuring Liberty

To configure Liberty for use with SPNEGO single sign-on, you have to create the Kerberos service principal name (SPN) and keytab file for the Liberty server that runs the i2 Analyze application, and edit the Liberty configuration to use SPNEGO single sign-on and the Active Directory registry.

Before you begin

To use SPNEGO single sign-on with the Liberty server that is deployed by i2 Analyze (and other Open Liberty servers), each server must use the same LTPA keys file. For more information about LTPA, see the LTPA section of the Open Liberty documentation for single sign-on.

The value that is set for the `ltpakeys.password` property in the `credentials.properties` file must match the password that is required to import the keys from the LTPA keys file. If you change the password in the `credentials.properties` file, you must redeploy i2 Analyze for the password change to take effect.

About this task

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Configure Liberty to use SPNEGO single sign-on by using the first two steps in Configuring SPNEGO authentication as a reference.
   a. Create the Kerberos SPN and keytab files on the domain controller.

      Note: Ensure that the host file on the Active Directory server uses the full host name, including the domain name, for the i2 Analyze server. Remove any entries that use only the short name for the i2 Analyze server. The value in the host file must match the value that is used for the SPN.
   b. Configure the server that hosts Liberty, and Liberty itself.
2. Configure Liberty to use the Microsoft Active Directory registry by using the instructions in Configuring LDAP user registries with Liberty as a reference. (Configuring Open Liberty is identical to configuring IBM WebSphere Liberty here.)
   a. Complete Step 1 to add the features to the `i2analyze\deploy\wlp\usr\servers\opal-server\server.xml` file.
   b. Complete Step 4 by using the Microsoft Active Directory Server example to populate the `<ldapRegistry>` element.

      Note: This information does not cover the configuration of Secure Sockets Layer (SSL) between Liberty and Active Directory. Do not include the `<ssl>` and `<keyStore>` elements from the example in your `server.xml` file.
   c. Ensure that the mapping between Active Directory and the i2 Analyze security schema is correct. Add the following code after the `<ldapRegistry>` element in the `server.xml` file:

      ```
      <federatedRepository>
        <primaryRealm name="">
          <participatingBaseEntry name=""/>
          <groupSecurityNameMapping inputProperty="cn" outputProperty="cn"/>
        </primaryRealm>
      </federatedRepository>
      ```

      Populate the empty `name` attribute values by using the following information:

- The <primaryRealm> element's name attribute has the same value as the realm attribute of the <ldapRegistry> element.
- The <participatingBaseEntry> element's name attribute has the same value as the baseDN attribute as the <ldapRegistry> element.

By default, all requests to access protected resources use SPNEGO authentication. If you previously deployed i2 Analyze with basic authentication, you must ensure that the basic registry is not present in the user.registry.xml file.

1. Using an XML editor, either remove or comment out the complete <basicRegistry> element in the i2analyze\deploy\wlp\usr\shared\config\user.registry.xml file.

Configuring the i2 Analyze application

Set the security administrator group to the value used in Microsoft Active Directory.

About this task

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using a text editor, open toolkit\configuration\environment\opal-server\environment-advanced.properties. Edit the value of the security.administrator.group property to the name of the group in Active Directory to use for administrators. For example, i2Admins.

   The mapping to the administrator group must be in terms of the Common Name that is defined in the <participatingBaseEntry> element of the <federatedRepository> element in the server.xml file. For more information about the <federatedRepository> element, see Configuring Liberty.
2. In a command prompt, navigate to the toolkit\scripts directory.
3. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
4. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```
5. Start Liberty:

   ```
   setup -t startLiberty
   ```

Testing i2 Analyze with SPNEGO single sign-on on the client workstation

To test the SPNEGO single sign-on setup, connect to i2 Analyze from a domain client workstation. The user that is logged in to the domain client is logged in to i2 Analyze.

Before you begin

Your web browser must be configured according to Step 3 in Configuring SPNEGO authentication.

You must be logged in to the client workstation as one of the users in the domain controller, who is in at least one group per security dimension in the i2 Analyze security schema.

About this task

Log in to the client workstation as users with different access levels. For each user, complete the following steps to demonstrate that authorization is working correctly when you are using SPNEGO single sign-on.

Procedure

1. Open your web browser, and navigate to `http://host_name/opal` (where *host_name* is the fully qualified domain name or IP address of the server that hosts the i2 Analyze application).
2. Create, browse, and search for data, to ensure that you are authenticated and authorized with the platform correctly.

   Note: When you create records, ensure that the permissions are set up so that you have access to view them.

What to do next

After you test your changes, you can configure user access to features. For example, to export a list of records in XLSX format, a user must be a member of a group that has the `i2:RecordsExport` permission under command access control. For more information, see Configuring command access control.

Configuring the Liberty user registry

Through the support in Liberty, i2 Analyze can use LDAP user registries such as Microsoft Active Directory to authenticate users. Especially in example or demonstration deployments, however, you can use Liberty's basic user registry to authenticate users and assign the system group memberships that they need.

Note: For instructions on configuring Liberty to use an LDAP registry, see Configuring LDAP user registries in Liberty.

Before you begin

If you are configuring the basic user registry to provide users in the early stages of the production deployment process, you must adjust it when you modify the groups in the security schema in the schema development or configuration development environments.

About this task

i2 Analyze users can have separate identifiers, user (principal) names, and display names. When you use the basic registry, the same value is used for all three purposes. You can use provisioning to modify users' display names in an example deployment, but in production you should favour one of the other approaches to authentication.

Procedure

1. Create the users and groups in Liberty for each of the group permissions elements in the security schema:

   a. In an XML editor, open the `user.registry.xml` file. You can find this file in the `C:\i2\i2analyze\deploy\wlp\usr\shared\config` directory of your Liberty installation.

   b. Use the following template to add your users and groups to the `user.registry.xml` file as the first child of the `<server>` element:

```
<basicRegistry id="basic" realm="WebRealm">
  <user name="" password="" />
    <group name="">
      <member name="" />
    </group>
</basicRegistry>
```

Use the following information to populate the template:

- There is a `<user>` element for each user of the system. The `<user>` element's `name` and `password` attributes must be populated for that user.
- There is a `<group>` element with a `name` attribute that matches the name of each user group mentioned in the security schema. You can also add groups that are not mentioned in the security schema.
- The `<group>` elements are populated by `<member>` elements. For a user to be a member of a group, a `<member>` element's `name` attribute must match that user's `name` attribute.

If you are using the example deployment, the user `Jenny` is a member of each group.

In the following example `user.registry.xml`, the users `Analyst1`, and `Clerk1` have been added into a subset of the groups. If you use the following example, log in as these users to see the different permission levels of each group:

```
<basicRegistry id="basic" realm="WebRealm">
  <user name="Jenny" password="{xor}FToxMSY="/>
  <user name="Analyst1" password="{xor}HjE+MyYsK24=" />
  <user name="Clerk1" password="{xor}HDM6LTRu" />
  <group name="Analyst">
    <member name="Jenny"/>
    <member name="Analyst1"/>
  </group>
  <group name="Clerk">
    <member name="Jenny"/>
    <member name="Clerk1"/>
  </group>
  <group name="Controlled">
    <member name="Jenny"/>
    <member name="Analyst1"/>
  </group>
  <group name="Unclassified">
    <member name="Jenny"/>
    <member name="Clerk1"/>
  </group>
  <group name="Administrator">
    <member name="Jenny"/>
  </group>
</basicRegistry>
```

2. Use the Liberty `securityUtility` command to encode the password for each user:
   a. Navigate to the `bin` directory of your Open Liberty deployment that is configured by the deployment toolkit. By default Liberty is deployed in the `C:\i2\i2analyze\deploy\wlp` directory.
   b. In a command prompt, run the following command:

      ```
      securityUtility encode password
      ```

The encoded password is displayed in the command line. Record the encoded password, including the `{xor}` prefix, and use the encoded password as the password in the `user.registry.xml` file.

For more information about using the security utility, see securityUtility command.

3. Save and close the file.

What to do next

To test that your changes have worked, log in to i2 Analyze as one of the users that you added to the user registry.

After you test your changes to the user registry, you can configure user access to features. For example, to upload records to the Information Store, a user must be a member of a group that has the `i2:RecordsUpload` permission under command access control. For more information, see Configuring command access control.

Configuring X.509 client certificate authentication with i2 Analyze

If your i2 Analyze deployment uses a user registry, you can enable it to use X.509 client certificate authentication. After successful configuration, users can log in to i2 Analyze with client certificates instead of user names and passwords.

After you configure client certificate authentication, a user does not need to enter a user name and password separately. Each certificate is associated with a single user to enable authentication. Anyone that has access to a client certificate can log in to i2 Analyze as the user associated with that certificate without entering a password.

To enable a user to log in using a client certificate, the client certificate must be installed in the user's personal certificate store on the workstation they are using to access i2 Analyze. After the client certificate is installed in the personal certificate store, the user can use the certificate to log in to i2 Analyze through Analyst's Notebook.

When a user connects using Analyst's Notebook, the user chooses the certificate to use to log in when they open a connection to the Information Store.

Intended audience

In the production deployment process, you might first configure client certificate authentication in the configuration or pre-production environments. As you move to a production deployment, you must replicate any configuration changes in any new deployments.

This information is intended for readers who are familiar with managing key databases and certificates, user authentication mechanisms, and the i2 Analyze toolkit.

Prerequisites

The starting point for configuring client certificate authentication is a deployment of i2 Analyze that is configured to use Transport Layer Security on connections to the HTTP Server, and between the HTTP Server and Liberty. For more information about configuring Transport Layer Security on connections to the HTTP Server, see Configuring Transport Layer Security with i2 Analyze.

Attention: i2 takes reasonable steps to verify the suitability of i2 Analyze for internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and

recommendations for protection of your systems. i2 accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

Client certificates

The client certificates that are used to authenticate users must be signed by a certificate authority that is trusted by the i2 Analyze server.

The common name in a client certificate must match a user name in the registry that i2 Analyze is using. A user that selects such a certificate logs in to i2 Analyze as the corresponding i2 Analyze user.

You can have as many client certificates as you require. Each certificate is associated with a single user in the registry. Each certificate can be installed on any number of workstations. Each workstation can have any number of certificates installed.

To demonstrate a working configuration, you can use a self-signed client certificate. For more information, see Creating a self-signed client certificate. However, in a production deployment you must use certificates that are signed by a certificate authority that is trusted by the i2 Analyze server.

There are many methods for obtaining an X.509 certificate that is signed by a certificate authority. When you receive a signed certificate, you also receive signer certificates so that you can trust the client certificates that are signed by that certificate authority. If the certificate authority that signed your certificates is not already trusted within the key database, you must add any signer certificates to the key database so that the certificate authority is trusted.

Creating a self-signed client certificate

The client certificate is used to log in and authenticate a user with i2 Analyze. Use the Java keytool utility to create a self-signed certificate.

About this task

Create a self-signed certificate to use as a client certificate to demonstrate a working configuration. If you are using client certificates that are signed by a certificate authority, you do not need to complete the following instructions.

Procedure

Create a keystore and self-signed certificate for a user by using the Java keytool utility.

1. Open a command prompt and navigate to the `i2analyze\deploy\java\bin` directory.
2. Create a keystore and certificate.

   For example, run the following command:

   ```
   keytool -genkeypair -alias "<username>" -keystore "C:\i2\i2analyze
   \<username>.p12" -dname "CN=<username>" -keyalg RSA -storepass
    "<password>"
   ```

   Important: Ensure that the value of `CN` matches the name of a user in the user registry for i2 Analyze. The user name cannot contain a comma (,).

   If you are using the example user registry, set the `<username>` to `Jenny`.

   Note: You will use the keystore to install the certificate on a client machine later.
3. Export the certificate from the keystore.

   For example, run the following command:

```
keytool -exportcert -alias "<username>" -keystore "C:\i2\i2analyze
\<username>.p12" -file "C:\i2\i2analyze\<username>.der" -storepass
 "<password>"
```

Note: You will import the certificate into the Liberty truststore later.

Configuring the truststore

To enable the i2 Analyze server to trust the client certificates, you must ensure that the signer of your client certificates is trusted within the Liberty truststore.

About this task

If you are using client certificates that are signed by a certificate authority, ensure that the certificate authority that signed the certificates is trusted within the Liberty truststore.

Procedure

The Liberty truststore must contain the certificates to ensure that the certificates received from the client are trusted.

1. Run the following command to import the required certificate into the truststore. If the truststore does not exist, it is created:

   ```
   keytool -importcert -alias "<signerKey>" -keystore "C:\i2\i2analyze\i2-
   liberty-truststore.p12" -file "C:\i2\i2analyze\<signer-certificate>.der" -
   storepass "<password>"
   ```

   Note: When you are using a self-signed client certificate, add the self-signed client certificate as a signer certificate. For example, `Jenny.der`.

Results

The truststore contains the signer certificates so that the client certificates can be trusted. The truststore is populated so that Liberty can use it to trust the client certificates.

Configuring i2 Analyze

To enable a user to log in using a client certificate, you must modify some of the configuration files for i2 Analyze.

About this task

Add a rewrite rule that enables client authentication on the IBM HTTP Server to the i2 Analyze configuration. Then, update the `web.xml` file for the application to enable client certificate authentication.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using a text editor, open the `configuration\environment\proxy\http-custom-rewrite-rules.txt` file. Add the following line between the `!Start_After_Rules!` and `!End_After_Rules!` lines to enable client certificate authentication:

   ```
   SSLClientAuth Optional
   ```

2. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

a. Add a child `<key-store>` element to the `<key-stores>` element.

For your truststore, specify the `type` as `trust-store` and `file` as the full path to your truststore.

For example, add the attributes as highlighted in the following code:

```
<application http-server-host="true" name="<opal-server>"
    host-name="<hostname>" secure-connection="true">
  ...
  <key-stores>
    <key-store type="key-store"
        file="C:/i2/i2analyze/i2-liberty-keystore.jks"/>
    <key-store type="trust-store"
        file="C:/i2/i2analyze/i2-liberty-truststore.jks"/>
  </key-stores>
  ...
</application>
```

b. Specify the truststore password in the credentials file. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file and enter a password for the truststore that you specified in the `topology.xml` file.

```
ssl.truststore.password=<password>
```

3. Use an XML editor to modify the `toolkit\configuration\fragments\opal-services\WEB-INF\web.xml` file.

Comment out the following lines so that form-based authentication is not used:

```
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>Form-Based Authentication Area</realm-name>
  <form-login-config>
    <form-login-page>/login.html</form-login-page>
    <form-error-page>/login.html?failed</form-error-page>
  </form-login-config>
</login-config>
```

In the login configuration section, add the following lines to define the client certificate authentication method:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>WebRealm</realm-name>
</login-config>
```

4. In a command prompt, navigate to the `toolkit\scripts` directory.

5. Stop Liberty:

```
setup -t stopLiberty
```

6. Update the i2 Analyze application:

```
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

8. Using an XML editor, open the `wlp\usr\servers\<opal-server>\server.xml` file.

a. Modify the `<ssl>` element with the id `defaultSSLConfig` to include `clientAuthenticationSupported="true"`.

For example:

```
<ssl clientAuthenticationSupported="true"
    id="defaultSSLConfig"
    keyStoreRef="defaultKeyStore"
    trustStoreRef="defaultTrustStore"/>
```

b. Modify the `<httpDispatcher>` element to include `trustedSensitiveHeaderOrigin="*"`.

For more information about the values that you can provide for the `trustedSensitiveHeaderOrigin` attribute, see HTTP Dispatcher (httpDispatcher).

For example:

```
<httpDispatcher enableWelcomePage="false"
    trustedSensitiveHeaderOrigin="*"/>
```

Results

The i2 Analyze application is configured to allow client certificate authentication.

Installing client certificates

After you create the client certificates, install the client certificates on the client workstations. After the client certificates are installed, they are available to be selected by users to use to log in to i2 Analyze.

About this task

On each client workstation that you want a user to be able to log in from, install the client certificate and exported keys as a Personal certificate. You can install multiple certificates and their associated keys on a workstation to allow multiple users to log in.

If you are not using a self-signed certificate, install the signer certificate that is associated with your client certificate so that your operating system can verify the client certificates.

Procedure

1. Copy your client certificate and the exported keys to a permanent directory on the client workstation.
2. Install the client certificate and exported keys to the **Personal** store:
   a. Double-click the keys file (for example, `Jenny.p12`). The **Certificate Import Wizard** is displayed.
   b. Click **Next**.
   c. Click **Browse**, and locate the keys file to import (for example, `Jenny.p12`). Then, click **Next**.
   d. Enter the password that you specified when the keys were exported from the key database, and click **Next**.
   e. Click **Place all certificates in the following store**.
   f. Click **Browse**, and select **Personal**.
   g. Click **Next**, and then click **Finish**.
3. If you are using a self-signed certificate, install the client certificate to the **Trusted Root Certification Authorities** store:
   a. Double-click the client certificate file (for example, `Jenny.der`).
   b. Click **Install Certificate**, and then click **Next**.

c. Click **Place all certificates in the following store**.
d. Click **Browse**, and select **Trusted Root Certification Authorities**.
e. Click **Next**, and then click **Finish**.
f. If the operating system cannot verify the certificate, a security warning is displayed.

   Click **Yes** to accept the certificate.

Results

The client certificate and exported keys are installed. Users can select a certificate to use to log in to i2 Analyze.

Testing the deployment

You can use X.509 certificates to authenticate users in i2 Analyze from Analyst's Notebook. The connection to the i2 Analyze server is secured, and the user is authenticated with the server by the client certificate that is provided by Analyst's Notebook.

Before you begin

A client certificate must be installed on the client workstation. For more information about installing the client certificates, see Installing client certificates.

The application server must be running.

About this task

Connect to the Information Store, and select a client certificate to use to authenticate with the i2 Analyze server.

Procedure

1. Enter the name of the repository, and the URL of the server to connect to.

   Use the HTTPS protocol to connect to the server. For example, `https://host_name/opal` (where `host_name` is the fully qualified domain name or IP address of the HTTP server).

   You are prompted to provide a certificate.
2. Select the client certificate to use to authenticate with the server from the list that is displayed.

Results

When client certificate authentication is configured correctly, you are logged in to i2 Analyze as the user associated with the selected certificate.

Configuring Liberty to use OCSP to check certificates

After you configure i2 Analyze to use client certificate authentication, you can configure Liberty to use Online Certificate Status Protocol (OCSP) to check the revocation state of the client certificates. For more information about OCSP, see Online Certificate Status Protocol.

About this task

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. In a text editor, open the `i2analyze\deploy\wlp\usr\servers\opal-server\jvm.options` file and add the following lines:

   ```
   -Dcom.sun.net.ssl.checkRevocation=true
   -Djava.security.properties=C:/i2/i2analyze/deploy/wlp/usr/servers/opal-server/java-security-ocsp.properties
   ```

   Where:

   - `checkRevocation` is set to `true` to instruct Liberty to check whether certificates have been revoked.
   - `java.security.properties` is the path to a properties file that contains the settings to configure OCSP.

2. In a text editor, create the `i2analyze\deploy\wlp\usr\servers\opal-server\java-security-ocsp.properties` file and add the following lines:

   ```
   ocsp.enable=true
   ocsp.responderURL=
   ```

   Where:

   - `ocsp.enable` is set to `true` to enable OCSP.
   - `ocsp.responderURL` is the URL of the OCSP service that is used to check the status of a certificate. When this value is specified, it overrides the value in the Authority Information Access extension on the certificate.

3. Restart Liberty:

   ```
   setup -t restartLiberty
   ```

What to do next

Log in to your deployment to test that revoked certificates are identified successfully.

If a user attempts to log in with a revoked certificate, a message is displayed in the Liberty logs. For example:

```
java.security.cert.CertPathValidatorException: Certificate has been revoked,
 reason: UNSPECIFIED, revocation date: Wed Jan 20 17:13:35 UTC 2021,
 authority: CN=ocsp, OU=i2, O=IBM, ST=England, C=GB, extension OIDs: []
```

If your OCSP service is unavailable, a message is displayed in the Liberty logs. For example:

```
The extended error message from the SSL handshake exception is: PKIX path
 validation failed: java.security.cert.CertPathValidatorException: Unable to
 determine revocation status due to network error
```

User provisioning

The process of *provisioning* users for i2 Analyze involves ensuring that users are members of the correct groups, and that the correct users can access the system. By default, when users are authenticated, they are provisioned from the information that the authentication mechanism holds.

If you're using a registry such as Microsoft Active Directory, then the user's identity, their name, and their group membership all come from that registry. If you're using an identity provider, then the information arrives in the token that the identity provider sends to i2 Analyze.

There are a few reasons why you might not want the default provisioning behavior in your system. For example, when you're using a registry, you might want to restrict the users who can access i2 Analyze

to a subset of the users in the registry. When you're using an identity provider, there are advantages to telling i2 Analyze about the users and groups who will use the system before they log in.

To perform custom provisioning, you have to edit the supplied configuration file according to your needs, and then tell i2 Analyze to use it.

Configuring user provisioning

Whether your deployment of i2 Analyze authenticates users through a registry or an identity provider, the basic procedure for custom user provisioning is the same. You must open the `provisioning-configuration.json` file, configure it, save it, and update the deployment.

Before you begin

To enable customization of user provisioning, you must change a setting in the `ApolloServerSettingsMandatory.properties` file. If you have not previously done so, set the `ProvisioningCompatibilityMode` property in that file to `false` and then redeploy Liberty.

Note: i2 Analyze sets the `ProvisioningCompatibilityMode` property to `true` during upgrade of deployments of version 4.4.3 and earlier that use federated user registries. In this mode, any member of the federated registry can log in to i2 Analyze, and the application only learns about users and groups when they authenticate for the first time.

Procedure

The provisioning configuration file controls how users from a user registry or an identity provider are provisioned in i2 Analyze.

1. Open the `configuration/live/provisioning-configuration.json` file in a text editor.
2. Use the information in The `provisioning-configuration.json` file to configure your custom provisioning settings.
3. Save the provisioning configuration file.
4. Update the deployment with your changes.

   The following steps deploy your changes without stopping the server by using a POST request to a REST endpoint.

   Note: To redeploy your changes by using only the deployment toolkit, see Redeploying Liberty. You must use the deployment toolkit if you're in a deployment with high availability, or you're deploying in your production environment.

   a. Update the configuration on the server:

      ```
      setup -t updateLiveConfiguration
      ```

   b. Update the running application through the `reload` endpoint:

      ```
      curl -i --cookie-jar cookie.txt -d j_username=user_name -d
       j_password=<password> http://<host_name>/<context_root>/
      j_security_check

      curl -i --cookie cookie.txt -X POST http://<host_name>/<context_root>/
      api/v1/admin/config/reload
      ```

      Note: To reload the server through the `reload` endpoint, you must have administrative access to i2 Analyze. For more information, see Using the admin endpoints.

      If your custom provisioning configuration mistakenly results in no users having administrative access, you can fix the problem and redeploy i2 Analyze by using the deployment toolkit.

Warning: The `reload` method updates the provisioning configuration without requiring a server restart, but any logged-in users might experience some disruption when you run it.

The `provisioning-configuration.json` file

The `provisioning-configuration.json` file comprises sections that enable you to configure the users and groups that are allowed to access the i2 Analyze system, and how the system responds to new users and groups.

The file has three possible sections, and you must always include two of them. The sections are:

- `registryConfig`
- `identityProviderConfig`
- `settings`

Registry configuration

If your deployment of i2 Analyze authenticates users through a registry, you can use custom provisioning to control which users and groups from the registry can access the system.

For details, see Custom provisioning with a user registry. If you're using an identity provider, you don't need to include the `registryConfig` object.

Identity provider configuration

If your deployment of i2 Analyze authenticates users through an identity provider, you can use custom provisioning to tell the system about the users and groups who will use the system before they log in.

If you don't use custom provisioning in these circumstances, the system provisions users based on the claims in the token that the identity provider sends to i2 Analyze.

For details, see Custom provisioning with an identity provider. If you're using a user registry, you don't need to include the `identityProviderConfig` object.

Settings

The `settings` object contains properties that define how changes in the user registry or identity provider are handled in i2 Analyze.

For example, if a user is added to the registry and the value of `processAddedUsers` is `true`, the user is provisioned for access when the system next reloads the registry, unless they don't pass the filters that you've defined. If the value is `false`, the user is not provisioned until you set it to `true` and update the system.

Note: If you're using an identity provider for authentication, most of these settings apply only when `provisionUsing` is set to `FILE`. For other values of `provisionUsing`, only the `defaultValuesForAddedGroups` setting has any effect.

`processAddedUsers`

A Boolean value that specifies whether to process new users that are added to the user registry or identity provider. The default value is `true`.

`processAddedGroups`

A Boolean value that specifies whether to process new groups that are added to the user registry or identity provider. The default value is `true`.

processChangedUsers

A Boolean value that specifies whether to process changed users in the user registry or identity provider. The default value is `true`.

processChangedGroups

A Boolean value that specifies whether to process changed groups in the user registry or identity provider. The default value is `true`.

processRemovedUsers

A Boolean value that specifies whether to process removed users from the user registry or identity provider. The default value is `true`.

processRemovedGroups

A Boolean value that specifies whether to process removed groups from the user registry or identity provider. The default value is `true`.

processEmptyGroups

A Boolean value that specifies whether to process empty groups in the user registry or identity provider. The default value is `true`.

defaultValuesForAddedGroups

An object that contains default settings for added groups. The object contains the following properties:

visibleToAllUsers

A Boolean value that specifies whether the group is visible to all users. The default value is `false`.

availableInSharing

A Boolean value that specifies whether the group is available in sharing. The default value is `false`.

reloadInterval

An object that contains settings that define how often the system reloads the user registry or identity provider to get the latest user and group information. The object contains the following properties:

value

An integer that specifies the reload interval. The default value is `30`.

units

A string that specifies the units of the reload interval. The default value is `MINUTES`.

enabled

A Boolean value that specifies whether the reload interval is enabled. The default value is `true`.

Examples

Example `provisioning-configuration.json` file for an identity provider:

```
{
  "identityProviderConfig": {
    "provisionUsing": "CLAIMS",
    "displayNameClaims": "name"
  },
```

```
  "settings": {
    "defaultValuesForAddedGroups": {
      "visibleToAllUsers": false,
      "availableInSharing": false
    }
  }
}
```

Example `provisioning-configuration.json` file for a user registry:

```
{
  "registryConfig": {
    "userFilters": [
      "*,ou=cambridge,*",
      "*,ou=administrators"
    ],
    "groupFilters": [
      "*"
    ],
    "userDisplayNameMappings": [
      {
        "fromSecurityName": "Jenny",
        "toDisplayName": "Jenny Jones"
      },
      {
        "from": "(.+)\\.(.+)@example.com",
        "to": "$1 $2"
      }
    ],
    "groupDisplayNameMappings": []
  },
  "settings": {
    "processAddedUsers": true,
    "processAddedGroups": true,
    "processChangedUsers": true,
    "processChangedGroups": true,
    "processRemovedUsers": true,
    "processRemovedGroups": true,
    "processEmptyGroups": true,
    "defaultValuesForAddedGroups": {
      "visibleToAllUsers": false,
      "availableInSharing": false
    },
    "reloadInterval": {
      "value": 30,
      "units": "MINUTES",
      "enabled": true
    }
  }
}
```

Custom provisioning with a user registry

When i2 Analyze is using a registry for user authentication, you can use custom provisioning to control which users and groups from the registry can access the system. To do so, you populate the `registryConfig` object in the `provisioning-configuration.json` file.

The `registryConfig` object contains the following properties:

`userFilters`

An array of strings that specify the user filters. Only users that match the filters can log in to i2 Analyze.

For example, the following filters mean that i2 Analyze will only accept users in the `ou=cambridge` and `ou=administrators` organizational units:

```
"userFilters": [
  "*,ou=cambridge,*",
  "*,ou=administrators"
]
```

`groupFilters`

An array of strings that specify the group filters. Only groups that match the filters can be used in i2 Analyze.

For example, the following filter means that all groups from the registry are available:

```
"groupFilters": [
  "*"
]
```

`userDisplayNameMappings`

An array of objects that allows you to customize the display names of users in i2 Analyze, on either an individual or a generalized basis.

For example, the first of the mappings below results in a user whose login name is "Jenny" being displayed as "Jenny Jones" in i2 Analyze. The second mapping uses a regular expression to extract the first and last names of users whose login names are email addresses in the `example.com` domain:

```
"userDisplayNameMappings": [
  {
    "fromSecurityName": "Jenny",
    "toDisplayName": "Jenny Jones"
  },
  {
    "from": "(.+)\\.(.+)@example.com",
    "to": "$1 $2"
  }
],
```

`groupDisplayNameMappings`

An array of objects that allows you to customize the display names of groups in i2 Analyze, in the same way as `userDisplayNameMappings`.

The value of the `to` output for each group is the value that i2 Analyze uses to determine a user's group membership while evaluating their command access control permissions, for example.

Custom provisioning with an identity provider

When i2 Analyze is using an identity provider for user authentication, you can choose whether provisioning happens "just in time", from the claims in the tokens that the identity provider sends, or

whether it is controlled by another file that you provide. The choice affects your users' experience of the system.

Just-in-time provisioning

Just-in-time provisioning is the default behavior of an i2 Analyze deployment that uses an identity provider for user authentication.

In just-in-time provisioning, i2 Analyze receives its information about a user and the groups they're a member of from claims in a token that the identity provider creates. In other words, i2 Analyze knows nothing about any users or groups until they appear in a claim. Rather, it builds up a picture as more users log in.

One of the consequences of this behavior is that users can't share artifacts with each other until they've both logged in at least once. It also means that in theory, any user with a valid token from the identity provider can log in to i2 Analyze.

To change the default behavior, you have to include the `identityProviderConfig` object in the `provisioning-configuration.json` file. The object contains the following properties:

`provisionUsing`

A string value that indicates the source of the identity provider's provisioning. Use `CLAIMS` to retain the default behavior but gain the ability to specify which claim contains the user's display name.

`displayNameClaims`

A string value that specifies the claim that contains the display name of the user. The default value is `name`. `displayNameClaims` is only used when `provisionUsing` is set to `CLAIMS`.

```
"identityProviderConfig": {
  "provisionUsing": "CLAIMS",
  "displayNameClaims": "name"
}
```

File-based provisioning

In file-based provisioning, you provide i2 Analyze with advance knowledge of the users and groups from the identity provider. This behavior improves on just-in-time provisioning in two ways:

- The users and groups that you specify in the file are the only ones that can access i2 Analyze or appear as system groups. This restriction is useful if you want to limit access to a subset of the users in the identity provider.
- The users and groups in the file are available as soon as the system starts, so users can share artifacts with each other before everyone has logged in.

To specify file-based provisioning, you must set the value of the `provisionUsing` property of the `identityProviderConfig` object to `FILE` in the `provisioning-configuration.json` file:

```
"identityProviderConfig": {
  "provisionUsing": "FILE"
}
```

This setting tells i2 Analyze to look for a file called `identity-provider-users-and-groups.json` in the same directory as the `provisioning-configuration.json` file. The file must contain the users and groups that you want to provision, in arrays named `users` and `groups`.

A user can only access the system if their user name is present in the `users` array. Their group membership is defined by the groups they are a member of in the `groups` array.

Note: To log into the system *successfully*, a user must receive at least the "Read only" access level for at least one value in every security dimension. For more information, see i2 Analyze security permissions.

`users`

In the `identity-provider-users-and-groups.json` file, the `users` array contains objects that specify the users available to the system.

`id`

A string that specifies the unique identifier of the user.

`userName`

A string that specifies the user name of the user.

`displayName`

A string that specifies the display name of the user.

`groups`

The `groups` array contains objects that specify the groups available to the system, and the users that are members of each group.

`id`

A string that specifies the unique identifier of the group.

`displayName`

A string that specifies the display name of the group.

`memberIds`

An array of strings that specify the unique identifiers of the users or groups that are members of the group.

An example `identity-provider-users-and-groups.json` file is shown below:

```
{
  "users": [
    {
      "id": "40bd47aa-a672-46d3-94b3-51c938aa5dd0",
      "userName": "Imogen.Evans@example.com",
      "displayName": "Imogen Evans"
    }
  ],
  "groups": [
    {
      "id": "6ec017b0-ba12-42d9-80e6-d99c37a0896b",
      "displayName": "Investigation Group 1",
      "memberIds": [
        "40bd47aa-a672-46d3-94b3-51c938aa5dd0"
      ]
    }
  ]
}
```

In this example, the user `Imogen Evans` is a member of the group `Investigation Group 1`.

Logging for provisioning

In new deployments of i2 Analyze, the provisioning process logs information to the console, and to the `console.log` and `i2_Analysis_Repository.log` files. This topic explains how to add provisioning logging to existing deployments, and how to configure the logging level.

Before you begin

The logging that you add or configure through this topic is for the provisioning mechanism that was introduced in i2 Analyze version 4.4.4. To see output, you must already have configured user provisioning in your deployment.

About this task

Adding provisioning logging to an existing deployment of i2 Analyze involves adding a new logger named `com.i2group.provisioning.UsersAndGroups` to the `log4j2.xml` file.

When provisioning logging is active, changing the logging level just requires changing the value of the `level` attribute.

After any change to the `log4j2.xml` file, you must redeploy the Liberty server so that the change takes effect.

Procedure

To add provisioning logging to an existing deployment of i2 Analyze that was upgraded to version 4.4.4 or later:

1. Using an XML editor, open the `log4j2.xml` file for the deployment. The file is in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.
2. Add the following `<Logger>` element to the file, alongside all the others:

   ```
   <!-- User and group provisioning -->
   <Logger additivity="false" level="INFO"
    name="com.i2group.provisioning.UsersAndGroups">
     <AppenderRef ref="APOLLOLOG"/>
     <AppenderRef ref="CONSOLE"/>
   </Logger>
   ```

   This element is configured identically to the one that new deployments of i2 Analyze have by default.
3. (Optional) To log more information about the provisioning process, change the value of the `level` attribute in the `<Logger>` element to `DEBUG`:

   ```
   <Logger additivity="false" level="DEBUG"
    name="com.i2group.provisioning.UsersAndGroups">
   ```

To apply your changes to the i2 Analyze application:

1. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
2. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```
3. Start Liberty:

   ```
   setup -t startLiberty
   ```

Note: If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Configuring JWT authentication

Starting with version 4.4.4, i2 Analyze supports JSON Web Token (JWT) authentication so that components in the same domain can interact with it on behalf of authenticated users. The Open Liberty server that hosts the i2 Analyze application uses this feature to enable interaction with the Connector Designer component. This topic explains how to configure Open Liberty for that purpose.

Before you begin

When you configure the Open Liberty application server to use JWT authentication, i2 Analyze detects the setting and issues JWT tokens in cookies to authenticated users. When those users make requests to components like Connector Designer, the client includes the cookie, and the JWT token is used to authenticate the user.

About this task

Open Liberty provides full support for JWT authentication, but in order for i2 Analyze to take advantage, you need to configure it in the `server.extensions.xml` file. The following procedure describes how to:

- Add the `jwt-1.0` feature to i2 Analyze
- Add a trust association interceptor that handles the cookies
- Create a keystore and a truststore for the certificates that sign the JWT tokens
- Configure the JWT builder and JWT consumer that manage the tokens

Procedure

Perform these steps to configure JSON Web Token authentication in your i2 Analyze deployment:

1. If you haven't already done so, stop the Liberty server:

   ```
   setup -t stopLiberty
   ```

2. Navigate to the `configuration\liberty\server.extensions.xml` file, and open it in an XML editor.
3. If the file already contains a `<featureManager>` element, add a new `<feature>` to it. Otherwise, add one just before the closing `</server>` element:

   ```
   <featureManager>
     <feature>jwt-1.0</feature>
   </featureManager>
   ```

   This element enables the `jwt-1.0` feature that provides access to JAVA API classes for creating, reading, and verifying JWT tokens.
4. Outside the `<featureManager>` element, add the trust association interceptor to the `server.extensions.xml` file:

   ```
   <trustAssociation invokeForUnprotectedURI="false"
                     id="JwtTAI" disableLtpaCookie="true">
     <interceptors invokeBeforeSSO="true"

    className="com.i2group.disco.container.security.internal.JwtTAI"
                   id="JwtTAI" invokeAfterSSO="false" enabled="true">
       <library id="JwtTAILib">
         <fileset dir="${shared.resource.dir}/security/tai"/>
       </library>
     </interceptors>
   ```

```
</trustAssociation>
```

5. After the `<trustAssociation>` element, add definitions of the JWT keystore and truststore to the file:

```
<keyStore id="jwtKeyStore" location="<JWT_KEYSTORE_LOCATION>"
 type="PKCS12" password="<JWT_KEYSTORE_PASSWORD>" />
<keyStore id="jwtTrustStore" location="<JWT_TRUSTSTORE_LOCATION>"
 type="PKCS12" password="<JWT_TRUSTSTORE_PASSWORD>" />
```

   The keystore will contain the private key for signing JWT tokens, while the truststore will contain the public key for verifying JWT tokens.

   For information about creating certificates for development using Java keytool, see Creating keystores, truststores, and certificates for development.

   The certificates that the Java keytool generates are self-signed and they should not be used in production deployments. When you are deploying into production, you must use certificates that are provided and signed by a trusted certificate authority.

6. Finally, add definitions of the JWT builder and consumer to the file:

```
<jwtBuilder id="i2-analyze-jwt-claims" audiences="i2-analyze"
            issuer="<i2-analyze-url>" signatureAlgorithm="RS256"
            keyStoreRef="jwtKeyStore" expiresInSeconds="120m" />

<jwtConsumer id="i2-analyze-jwt-claims" audiences="i2-analyze"
             issuer="<i2-analyze-url>" signatureAlgorithm="RS256"
             trustStoreRef="jwtTrustStore" clockSkew="5m" />
```

   These settings must be the same on all Liberty servers in the i2 Analyze deployment. In particular:

   - The `id`, `audiences`, and `signatureAlgorithm` attributes must be set as above.
   - Set `issuer` to the URL of the i2 Analyze deployment. For example, `https://i2-analyze:9443`.
   - Set `keyStoreRef` and `trustStoreRef` to the identifiers of the keystore and truststore that you created earlier.
   - i2 suggests the `expiresInSeconds` and `clockSkew` values as reasonable starting points.

   For more information about the attributes and their values, see the Open Liberty documentation.

   If you do change the `expiresInSeconds` setting, bear in mind that smaller values cause users to re-authenticate more frequently, while larger values increase the risk from stolen tokens, which remain valid for longer.

7. Save and close `server.extensions.xml`, and then redeploy and restart the Liberty server:

```
setup -t deployLiberty
setup -t startLiberty
```

Controlling access to features

You can control access to the features and types of command that are available to i2 Analyze users. To restrict (and in some cases, to allow) access to features, you need to specify a command access control file.

Note: In the production deployment process, you first configure access to features in the configuration development environment.

You can use a command access control file to specify which permissions to assign to the user groups that were allocated during authentication (the *system groups*). The command access control file contains permissions that apply to i2 Analyze and, where applicable, to clients and client services.

Behavior with no command access control file

When you create an example deployment, i2 Analyze is configured with the command access control file that the example contains. If *no* file is specified in the configuration, access control is not enabled and all users have permission to use the same set of features:

- Create, edit, and delete Information Store records though i2 Analyst's Notebook
- Create, retrieve, edit, and delete Analyst's Notebook charts in the Chart Store
- Create and read notes on records and charts
- Create, edit, and save artifacts such as Visual Queries
- Interact with external data sources through i2 Connect connectors

Note: Feature availability also depends on support in the i2 Analyze deployment. For example, users can't use connectors if the deployment doesn't include the i2 Connect gateway.

Behavior with a command access control file

In contrast, a command access control file *must* be present in order for users to receive permission to use the following features:

- Export search results to a CSV file, or a list of records to an XLSX file
- Use the i2 Notebook web client
- Upload i2 Analyst's Notebook charts to the Chart Store in bulk
- Share records and artifacts such as saved Visual Queries with other users
- Create alerts through the REST API
- Create and manage custom user groups
- Administrator permissions

When you specify a command access control file, access to *all* features is controlled by that file. Before they can use a feature, you must give the appropriate permission to a group of users.

When you upgrade a system that has access to specific features enabled, ensure that you check for new features that require new permissions. Without updating your file to add permissions, access to the features is denied to all users. For information about any new permissions, see Configuration and database changes.

Configuring command access control

You can use command access control to determine which commands and features users can access. You can create a command access control file to match the specific needs of your deployment.

About this task

In any command access control file that you create, the group names in the file must match the names of system user groups from the user registry.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

Create and configure the command access control file.

1. Create a command access control file.
   a. Navigate to the directory in the deployment toolkit that contains the example security schema: `toolkit\configuration\examples\security-schema`.
   b. Copy the `example-command-access-control.xml` file to the `configuration\fragments\opal-services\WEB-INF\classes` directory, and rename it to `command-access-control.xml`.
2. Modify the command access control file.
   a. Open the `command-access-control.xml` file in your XSD-aware XML editor. For more information, see Setting up your XSD aware XML editor. The associated XSD file is `toolkit\scripts\xsd\CommandAccessControl.xsd`.
   b. Use the reference information to specify the access control that your system requires.
   c. Save the completed file.
3. Set the command access control file to be used in the deployment.
   a. Using a text editor, open the `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties` file.
   b. Specify your command access control file as the value for the `CommandAccessControlResource` property. For example:

      ```
      CommandAccessControlResource=command-access-control.xml
      ```
   c. Save the file.

Redeploy i2 Analyze to update the application with your changes.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```
4. Start Liberty:

   ```
   setup -t startLiberty
   ```

What to do next

Connect to your deployment and test that members of each user group have the correct access to features. Continue to change the configuration until you are satisfied with the access of each user group.

After you set command access control, you can revert to the default state by ensuring that the `CommandAccessControlResource` property in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties` file has no value.

Enabling access to the i2 Notebook web client

The default set of command access control permissions do not provide access to the i2 Notebook web client. If you have a deployment of i2 Analyze that includes the Chart Store or the Information Store, and

you want users to be able to use the i2 Notebook web client, you must specify it in a command access control file.

Before you begin

As well as command access control permissions, the i2 Notebook web client requires a secure connection between the user's web browser and the i2 Analyze server. For information on setting up a secure connection to the i2 Notebook web client, see Configuring Liberty for TLS.

About this task

By default, when users log in to i2 Analyze through their web browser, they see the i2 Investigate web client. To change that behavior, you must give specific access to the i2 Notebook web client to some or all of your users.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

To provide access to the i2 Notebook web client to users in some or all system user groups, you must add the `i2:Notebook` command access permission for the groups in question.

1. If you have not already done so, follow the instructions in Configuring command access control to set up a command access control file in your deployment of i2 Analyze.
2. In a text editor, modify the command access control file:
   a. Open the file, which is conventionally named `command-access-control.xml` and found in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes` directory.
   b. Add permission to access the i2 Notebook web client to one of the user groups. For example:

```
<CommandAccessPermissions UserGroup="Analyst">
  <Permission Value="i2:RecordsUpload"/>
  <Permission Value="i2:RecordsDelete"/>
  <Permission Value="i2:RecordsExport"/>
  <Permission Value="i2:ChartsUpload"/>
  <Permission Value="i2:ChartsDelete"/>
  <Permission Value="i2:Notebook"/>
</CommandAccessPermissions>
```

3. Save and close the `command-access-control.xml` file.
4. In a command prompt, navigate to the `toolkit\scripts` directory.
5. Stop Liberty:

```
setup -t stopLiberty
```

6. Update the i2 Analyze application:

```
setup -t deployLiberty
```

7. Start Liberty:

```
setup -t startLiberty
```

Enabling access to the Server Admin Console

The i2 Analyze Server Admin Console enables administrators to perform a range of configuration and management tasks from a web browser. The console supports different tasks through a set of apps, and you can use command access control to provide users with access to those apps as required.

Before you begin

At this version of i2 Analyze, the admin console can include the following apps:

- **i2 Connect gateway status** allows users to view the status of the i2 Connect gateway, to preview the configured services, and to reload the gateway.
- **i2 Analyze type conversion** allows users to create type conversion mappings between connector, gateway, and Information Store schemas.
- **User group management** allows users to create, edit, and delete custom user groups, and to configure the sharing settings of all user groups.

Users who are members of system groups that have the `i2:Administrator` command access control permission can access all the apps in the admin console.

Users who are members of system groups that have the `i2:Administrator:Groups` permission can access the user group management app.

Users who are members of system groups that have the `i2:Administrator:Connectors` permission can access the gateway status and type conversion apps.

About this task

To enable access to the admin console for any particular user, you must ensure that they're a member of a system group that has the appropriate command access control permission. This task describes how to configure and deploy the necessary settings.

Procedure

To arrange or verify that the right groups have the right command access control permissions, follow these steps:

1. If you have not already done so, follow the instructions in Configuring command access control to set up a command access control file in your deployment of i2 Analyze.
2. In an XML editor, modify the command access control file:
   a. Open the file, which is conventionally named `command-access-control.xml` and found in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes` directory.
   b. Check that at least one system group has the `i2:Administrator` permission. For example, if you have a system group named "Administrator" that should have access to all administrative functions, you might have this setting:

      ```
      <CommandAccessPermissions UserGroup="Administrator">
         <Permission Value="i2:Administrator" />
      </CommandAccessPermissions>
      ```

   c. Add the `i2:Administrator:Groups` permission to a system group that you want to have access to the user group management app.

      For example, if you wanted a group of senior analysts to have access, you might add the following setting:

      ```
      <CommandAccessPermissions UserGroup="SeniorAnalyst">
      ```

```
      ...
      <Permission Value="i2:Administrator:Groups" />
   </CommandAccessPermissions>
```

   d. Similarly, add the `i2:Administrator:Connectors` permission to a system group that you want to have access to the gateway status and type conversion apps.

3. Save and close the file.

Then, to arrange or verify that a user is a member of one of the groups with permission to access the admin console, follow these steps:

1. Examine the membership lists of the groups that have the administrator permissions.

   If you're using the basic Liberty user registry, you can open the file at `i2\i2analyze\deploy\wlp\usr\shared\config\user.registry.xml`.

2. If necessary, add the user to the appropriate group.

   Again, if you're using the Liberty registry, you might have a user named "Jenny" who's a member of the "Administrator" system group:

   ```
   <group name="Administrator">
      ...
      <member name="Jenny"/>
   </group>
   ```

   Or a user named "Imogen" who's a member of the "SeniorAnalyst" system group:

   ```
   <group name="SeniorAnalyst">
      ...
      <member name="Imogen"/>
   </group>
   ```

Finally, if you made changes to the `command-access-control.xml` file above, redeploy and restart the Liberty server:

1. At a command prompt, navigate to the `toolkit\scripts` directory.
2. Run the following commands:

   ```
   setup -t deployLiberty
   setup -t startLiberty
   ```

What to do next

Verify access to the admin console by opening a web browser and navigating to `<i2-Analyze-URL>/admin`, where `<i2-Analyze-URL>` is the URL used to access your i2 Analyze deployment.

When you log in as a user who is a member of a group with the appropriate command access control permissions, you should see the apps that are available to you.

For more information about command access control, see Controlling access to features.

The command access control file

The structure of the command access control XML configuration file and the permissions that it contains combine to control which i2 Analyze commands and features are available to users. Use the following reference and example information when you create your own configuration file.

- File structure
- Command access permissions

File structure

<CommandAccessControl>

The <CommandAccessControl> element is the root of the configuration file. It contains child <CommandAccessPermissions> elements.

<CommandAccessPermissions>

The <CommandAccessPermissions> element contains the access permissions for groups of users.

The UserGroup attribute specifies the user group that the access permissions apply to. The value of the attribute must match the display name of a user group. To specify that the permissions apply to all user groups, you can use the * wildcard.

The <CommandAccessPermissions> element contains one or more child <Permission> elements.

<Permission>

The Value attribute of the <Permission> element defines a permission that members of the user group that is specified in the parent <CommandAccessPermissions> element have access to.

For the list of values that you can specify for the Value attribute, see Command access permissions

The following example allows users in all system groups to upload records and charts, and members of the "Analyst" system group to delete records and charts too:

```
<tns:CommandAccessControl
    xmlns:tns="http://www.i2group.com/Schemas/2018-01-19/
CommandAccessControl"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.i2group.com/Schemas/2018-01-19/
CommandAccessControl CommandAccessControl.xsd">

  <CommandAccessPermissions UserGroup="*">
    <Permission Value="i2:RecordsUpload"/>
    <Permission Value="i2:ChartsUpload"/>
    <Permission Value="i2:Notes"/>
  </CommandAccessPermissions>

  <CommandAccessPermissions UserGroup="Analyst">
    <Permission Value="i2:RecordsDelete"/>
    <Permission Value="i2:ChartsDelete"/>
  </CommandAccessPermissions>

</tns:CommandAccessControl>
```

Command access permissions

The i2 Analyze command access permissions control access to features across a number of categories:

- Record and chart permissions control access to commands for record and chart management.
- Web client permissions control access to features in the web client.
- Connector permissions control access to connectors when your deployment includes the i2 Connect gateway.
- Sharing permissions control access to commands for sharing artifacts with other users.

- Administrator permissions control access to REST API endpoints (including the admin endpoint), and to the apps within the admin console.

Record and chart permissions

`i2:RecordsUpload`

Members of groups that have this permission can create and modify records and upload them to the Information Store.

Without this permission, users can search for records and add them to charts, but cannot upload changes to records.

`i2:RecordsDelete`

Members of groups that have this permission can delete records that were originally uploaded through Analyst's Notebook.

Without this permission, users can search for records and add them to charts, but cannot delete records from the Information Store.

`i2:RecordsExport`

Members of groups that have this permission can:

- Export records that are returned in search results to a CSV file
- Export a list of records in the i2 Notebook web client to an XLSX file
- Copy a list of records in the i2 Notebook web client to the clipboard

Without this permission, users cannot perform any of the above operations.

`i2:ChartsRead`

Members of groups that have this permission can search for and retrieve charts from the Chart Store.

Without this permission, users cannot search for or retrieve charts.

`i2:ChartsUpload`

Members of groups that have this permission can create and modify Analyst's Notebook charts and upload them to the Chart Store. Modifying a chart includes deleting versions of a chart, but not deleting the chart itself.

Without this permission, users can save Analyst's Notebook charts locally, but cannot upload new charts and modifications to existing charts.

Note: This permission automatically includes the `i2:ChartsRead` permission. You do not need to give both permissions to the same user groups.

`i2:ChartsBulkUpload`

Members of groups that have this permission receive access to the **Upload from Folder** feature in i2 Analyst's Notebook that enables users to upload charts from disk to the Chart Store in bulk.

Note: This permission automatically includes the `i2:ChartsUpload` permission. You do not need to give both permissions to the same user groups.

`i2:ChartsDelete`

Members of groups that have this permission can delete charts that were originally uploaded through Analyst's Notebook.

Without this permission, users cannot delete charts from the Chart Store.

`i2:Notes`

Members of groups that have this permission can create and access notes on records and charts.

Without this permission, notes are not displayed in the Notes tab, and the contents of any notes are not searchable.

`i2:CustomTypes`

Members of groups that have this permission can create, delete, and edit custom entity, link, and property types in Analyst's Notebook charts. The functionality that allows users to perform these actions is available through palettes, the Record Inspector, and the import wizard.

Without this permission, user interface elements for manipulating custom types are not displayed. However, users can still *use* custom types in charts that other users have created.

Web client permissions

`i2:Notebook`

Members of groups that have this permission can access the i2 Notebook web client. Members of groups without this permission see the i2 Investigate web client instead.

For more information, see Enabling access to the i2 Notebook web client.

Connector permissions

`i2:Connectors`

If you are using the i2 Connect gateway, members of groups that have this permission can view all i2 Connect connectors.

Without this permission, i2 Connect connectors are not visible unless individual connectors are specified by using the `i2:Connectors:<connector-id>` permission.

`i2:Connectors:<connector-id>`

If you are using the i2 Connect gateway, members of groups that have this permission can view the i2 Connect connector with the matching `<connector-id>`. For example, `i2:Connectors:example-connector`.

Without this permission, the specified i2 Connect connector is not visible.

Sharing permissions

`i2:Share`

Members of groups that have this permission receive all the permissions listed below. They can take advantage of all the features for sharing records and artifacts that i2 Analyze supports.

Without this permission, users can have artifacts shared with them, but cannot themselves share unless they have one of the more specific permissions.

`i2:Share:Artifacts`

Members of groups that have this permission can share artifacts with specific users and groups.

Members of groups that have the `i2:Administrator:SavedArtifacts` permission can share artifacts with anyone, regardless of whether they also have this permission.

`i2:Share:Records`

Members of groups that have this permission can share records with other i2 Analyze users, and see the records that other users have shared.

Administrator permissions

`i2:AlertsCreate`

Members of groups that have this permission can access the REST API alerts endpoint to create and send alerts to i2 Analyze users. For more information, see Managing i2 Analyze.

`i2:Administrator`

Members of groups that have this permission can access the REST API admin endpoints and all the apps in the admin console. They also get the privileges of all the other administrator permissions.

For more information, see Using the admin endpoints and i2 Analyze Server Admin Console.

`i2:Administrator:Connectors`

Members of groups that have this permission can access the REST API endpoints for managing connectors. They can also access the i2 Connect gateway status and i2 Analyze type conversion apps in the admin console.

`i2:Administrator:Groups`

Members of groups that have this permission can access the User group management app in the admin console.

`i2:Administrator:Indexing`

Members of groups that have this permission can access the REST API endpoints for viewing the status of the Information Store index, and for clearing and rebuilding it if necessary.

`i2:Administrator:SavedArtifacts`

Members of groups that have this permission have unrestricted access to all saved artifacts, including the ability to change who they're shared with. They can also share any new artifacts of their own.

Item type security

In some deployments of i2 Analyze, it is necessary to configure the system so that only users in specific system groups are able to interact with records with particular item types. For users without the necessary permissions, records with those types are invisible - it is as if they do not exist.

The `type-access-configuration.xml` file in the `toolkit/configuration/live` directory contains the item type security configuration, which defines how system user groups have visibility of specified item types.

Note: The `type-access-configuration.xml` file allows item type security to be determined from user membership of system groups. However, it is possible to configure item type security based on additional user information, for example their record security access permissions (defined by the security schema). To do this, you must implement the item type security SPI.

By default (and ignoring metadata), the `type-access-configuration.xml` file contains only its root element:

```
<tns:TypePermissions>
</tns:TypePermissions>
```

If the file does not exist or remains in its default state, then all users have access to all records, subject to the rules of the security schema.

Note: The configuration file is processed after item type conversion takes place. As such, the restrictions you define apply only to the types that are present post-mapping.

Configuration example

Given a deployed schema that contains the entity types `ET1`, `ET2`, `ET3`, and `LT1`, consider the following configuration:

```
<tns:TypePermissions DefaultSchemaShortName="...">
  <ItemType Id="ET1" SchemaShortName="...">
    <Allow>
      <UserGroup Name="Analyst"/>
      <UserGroup Name="Clerk"/>
    </Allow>
  </ItemType>
  <ItemType Id="ET3">
    <Allow></Allow>
  </ItemType>
  <ItemType Id="LT1"></ItemType>
</tns:TypePermissions>
```

This configuration defines the following:

1. Records with type `ET1` are visible only to users who belong to the `Analyst` or `Clerk` system groups. Users belonging to other groups are not able to see records with this type.
2. There is no permission for the type `ET2`. All users can see records with that type.
3. The permission for type `ET3` contains an empty `<Allow>` element. This means that records with type `ET3` are invisible to *all* users except those in groups that have the `i2:Administrator` command access permission.
4. The permission for type `LT1` contains *no* `<Allow>` element. As such, just like `ET2`, all users can see records with the `LT1` type.

Schema short names

In a deployment of i2 Analyze that contains several schemas, it is possible for item types from different schemas to have the same identifier. To avoid this potential problem, you can specify the short name of the schema that contains each type you want to configure.

As shown in the example above, you specify the schema short name through either the `SchemaShortName` attribute on an `<ItemType>` element, or the `DefaultSchemaShortName` attribute on the root `<tns:TypePermissions>` element.

If you omit the `SchemaShortName` attribute for an `<ItemType>` element, the value of `DefaultSchemaShortName` is used instead. If you provide neither attribute, i2 Analyze attempts to resolve the identifier against all schemas in the deployment. If the item type appears in more than one schema (or in none of them), i2 Analyze logs a warning that it cannot resolve the specified item type.

Note: It is not possible to specify more than one `<ItemType>` element with the same `Id` and `SchemaShortName` attributes.

Other considerations

Take note of the following considerations when you configure item type security in your deployment of i2 Analyze.

Command access control

Users in groups with the `i2:Administrator` command access permission are never affected by item type permissions.

Connectors & services

A connector service is hidden from the list of services on the client if *all* the possible item types of result records, as defined by `resultItemTypeIds` in the service configuration, are invisible to the user. If all of a connector's services are hidden for this reason, the connector itself is hidden.

Entities & links

Item type security restrictions on one entity type have no implications for other entity types. Similarly, restrictions on one link type have no implications for other link types, or for entity types. However, restrictions on the visibility of entity records *can* also affect the visibility of link records.

Taking the example configuration above, the deployed schema might specify that the entity type `ET1` is the *only* permitted type for records at one end of links with type `LT1`. In other words, `ET1` is the only entity type identifier in the `fromTypeIds` or `toTypeIds` attribute for the link type, like this:

```
<LinkType Id="LT1" FromTypeIds="ET1" ToTypeIds="ET2" ...>
```

In this situation, users who cannot see records of type `ET1` also cannot see records of type `LT1`.

On the other hand, if the definition of `LT1` specifies other valid link end types that are not restricted for the same users, as in this case, then records of type `LT1` remain visible:

```
<LinkType Id="LT1" FromTypeIds="ET1 ET2" ToTypeIds="ET2" ...>
```

Seeds

Item type security restrictions can affect the behavior of seeded search services. The following rules apply to services configured with `seedConstraints`:

- If the service accepts seeds of a restricted type, and does not specify that it must have at least one seed of that type (that is, it is possible to use the service without a seed of the restricted type), then *that constraint* is removed from the user's view.
- If the service accepts seeds of a restricted type, and it must have at least one seed of that type, then *the service* is removed from the user's view.

Item type conversion

As previously mentioned, item type restrictions apply only after item type conversion has taken place.

Asynchronous queries

Asynchronous queries are validated when they are launched, to ensure that the seeds are visible to the user.

If a user's permissions change after initiating the asynchronous query on the connector, the query runs to completion, but any results no longer visible to the user are filtered out of the result set (which might be empty as a result).

Changing item type permissions

By default, deployments of i2 Analyze do not define item type permissions. All users can see records of all types, provided that per-record security does not prevent them from doing so. To define or subsequently modify item type permissions, you must edit and redeploy the type access configuration.

Before you begin

The type access configuration is one of a group of configuration files that you can modify and send to the i2 Analyze server without the need for system downtime. These files are stored in the `toolkit/configuration/live` directory.

To enable updating the server without the need to restart it, ensure that you have access to a command-line tool such as `postman` or `curl`.

Procedure

1. Edit the configuration file.
   a. If you do not already have one, obtain and configure an XML editor.
   b. In the XML editor, open the `toolkit/configuration/live/type-access-configuration.xml` file.
   c. Using the reference and example information, modify the file to define the type access permissions that you need.
2. Update the deployment with your changes.

   The following method deploys your changes without stopping the server, through a POST request to a REST endpoint.

   To redeploy your changes using only the deployment toolkit, see Redeploying Liberty.

   You must use the deployment toolkit if you are in a deployment with high availability, or if you are deploying to your production environment.

   a. At the command line, navigate to the `toolkit/scripts` directory.
   b. Update the server with your configuration file:

      ```
      setup -t updateLiveConfiguration
      ```

   c. Update the running application by using the `reload` endpoint. Make sure that you provide the credentials of a user with administration rights:

      ```
      curl -i --cookie-jar cookie.txt -d j_username=<user_name>
                                      -d j_password=<password>
           http://<host_name>/<context_root>/j_security_check

      curl -i --cookie cookie.txt -X
           POST http://<host_name>/<context_root>/api/v1/admin/config/reload
      ```

   Warning: `reload` updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

   The server validates the item type configuration as it loads, and returns any errors in its response.
3. Test the new and updated item type permissions.

   A good way to verify that your item type security configuration is loaded correctly is to call connector or gateway schema endpoints and search the response body for an item type that the current user should not be able to see.

The item type security configuration file

The `type-access-configuration.xml` file controls which users get access to each of the item types declared in the schema. The set of permitted XML elements in an item type security configuration file is relatively small.

Root element

`<TypePermissions>`

`<TypePermissions>` is the root element of the item type security configuration file. In the file that deployments of i2 Analyze receive by default, the element is empty and its name is prefixed with the `tns` namespace:

```
<tns:TypePermissions DefaultSchemaShortName="...">
  ...
</tns:TypePermissions>
```

The `<TypePermissions>` element has a single, optional attribute named `DefaultSchemaShortName`. If the item types that you want to constrain are all or mostly defined in a particular schema, then it is efficient to provide the short name of that schema here.

When the `<TypePermissions>` element is empty, there are no item type constraints on the records that users can see.

Item type elements

`<ItemType>`

The `<TypePermissions>` root element supports any number of child `<ItemType>` elements that specify the type security permissions. `<ItemType>` is the only permitted child of `<TypePermissions>`:

```
<TypePermissions>
  <ItemType Id="..." SchemaShortName="...">
    ...
  </ItemType>
  ...
</TypePermissions>
```

The `<ItemType>` element has two attributes: `Id`, which is mandatory; and `SchemaShortName`, which is optional:

- `Id` is the identifier of the item type, as defined in the schema that contains it.
- `SchemaShortName` is the short name of the schema that contains the item type. When this attribute is set, it overrides `DefaultSchemaShortName` in the parent element.

Each item type for which the file contains permissions appears in exactly one `<ItemType>` element. If an `<ItemType>` element is empty, however, it is as if that element does not exist.

`<Allow>`

The item type security model assumes that if you want to control access to a particular type, then usually you want to make it so that only users in particular system groups can see records that have that type.

The `<ItemType>` element supports a single `<Allow>` child element. As soon as you add the element, access to the type is denied to all groups that are not specifically mentioned:

```
<TypePermissions>
  <ItemType Id="...">
```

```
    <Allow>
      ...
    </Allow>
  </ItemType>
  ...
</TypePermissions>
```

The `<Allow>` element has no attributes. If an `<ItemType>` element has an empty `<Allow>` child element, then only users who have the `i2:Administrator` command access permission can see records of that type.

`<UserGroup>`

The `<Allow>` element supports any number of child `<UserGroup>` elements. Members of each system user group that you specify (as well as users who have the `i2:Administrator` command access permission) are allowed to see records that have the parent item type:

```
<TypePermissions>
  <ItemType Id="...">
    <Allow>
      <UserGroup Name="..."/>
      ...
    </Allow>
  </ItemType>
  ...
</TypePermissions>
```

The `<UserGroup>` element has a single, mandatory `Name` attribute. For each system user group that should have permission to see records of the specified type, the `<Allow>` element must contain a `<UserGroup>` element whose `Name` attribute is set to the name of the user group.

Changing how item type access is determined

In some deployments of i2 Analyze, it is necessary to configure the system so that users are only able to access and interact with records with particular item types. By default, i2 Analyze determines which item types a user can access from their membership of system user groups, using the `type-access-configuration.xml` file.

If this suits your needs, see Item type security to learn how to configure users' access to item types in this way. To determine which item types a user can access based on other user information, you must implement the item type security SPI.

By implementing the item type security SPI, user item type access can be determined by any combination of:

- User principal name
- User system group membership
- User security dimension access permissions (defined by the i2 Analyze security schema)

The SPI is a set of Java interfaces that you can implement with your own Java classes. Then, you can configure i2 Analyze to use your implementation instead of the default one.

Implementing the SPI

See the `com.i2group.disco.security.spi` package in the i2 Analyze Security SPI documentation for details, but the two interfaces that you need to implement are:

- `IUserItemTypeAccessResolver`. This interface is used to get the access levels assigned to a specific user for each item type.

- `IUserItemTypeAccessResolverProvider`. This is the interface that i2 Analyze uses to obtain `IUserItemTypeAccessResolver` instances for specific users. It also exposes a method that allows you to validate and reload the item type security configuration on server startup and reload.

You must package your SPI implementation into a single JAR file.

Configuring i2 Analyze to use your implementation

Note: Customizing item type access resolution should take place during the development phase of your deployment, in the configuration development environment. See Deployment phases and environments for more information.

For i2 Analyze to use your item type security implementation, you must:

1. Copy the JAR file containing your implementation into the `toolkit/configuration/fragments/opal-services/WEB-INF/lib` i2 Analyze toolkit directory where it will be deployed into the application.
2. Instruct i2 Analyze to use your implementation by setting the `IUserItemTypeAccessResolverProvider` property in the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file to the fully qualified class name of your `IUserItemTypeAccessResolverProvider` implementation. For example:

   ```
   IUserItemTypeAccessResolverProvider=com.example.ImplementationClassName
   ```
3. Redeploy and restart liberty. From the `toolkit/scripts` directory, run:

   ```
   setup -t stopLiberty
   setup -t deployLiberty
   setup -t startLiberty
   ```

High availability deployments

For high availability and disaster recovery deployment topologies, performing the above procedure in the configuration development environment ensures that your SPI implementation is copied to each Liberty server when you configure the pre-production environment.

To make changes to the SPI implementation in a pre-production or test environment that has multiple Liberty servers, you must ensure that the updated JAR file is copied to the toolkit of each Liberty server.

Displaying information to users

i2 Analyze provides two mechanisms for displaying information to users when they connect to i2 Analyze.

- You can display a privacy prompt to users that they must accept before they can connect to the server.
- You can display a welcome page in Analyst's Notebook with information about your deployment.

Displaying a welcome page in Analyst's Notebook

When users log in to your deployment of i2 Analyze from Analyst's Notebook, you can arrange for them to see a welcome page that introduces it. The page gets its contents from an HTML file that is entirely under your control.

Before you begin

The information that users see in a welcome page is at your discretion, but typical content that you might present includes:

- Your corporate branding, such as a logo or a color scheme
- A description of *your* i2 Analyze deployment, including the features that it provides
- Links to sections of the i2 documentation that explain the features you deployed
- Links to *your* documentation that gives detail on using i2 Analyze in your organization

About this task

To set up a welcome page, you replace the contents of an HTML file in the deployment toolkit with the information that you want to provide, and then configure the server to request Analyst's Notebook to display it.

Procedure

Complete the following steps to create and configure a welcome page:

1. Modify the `toolkit/configuration/fragments/common/welcomepage.html` file to contain information about your deployment of i2 Analyze.
2. Configure i2 Analyze to enable the welcome page. In the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file, set the `EnableWelcomePage` property to `true`:

   ```
   EnableWelcomePage=true
   ```
3. Run the following commands from the `toolkit/scripts` directory on your Liberty server to update the application with your changes:

   ```
   setup -t stopLiberty
   setup -t deployLiberty
   setup -t startLiberty
   ```

Results

The next time that a user logs in to i2 Analyze from Analyst's Notebook, they will see your welcome page.

The dialog that hosts the welcome page also contains a checkbox that users can select if they don't want to see the welcome page again. To reverse this setting, users can reopen the welcome page from the User menu and clear the checkbox.

Configuring a privacy prompt

Clients that connect to i2 Analyze can display a prompt to users that they must accept before they can connect to the server. If the user rejects the prompt, they cannot connect to i2 Analyze.

About this task

To set up a privacy prompt, you must provide the content of the prompt that your users are required to accept and how often they are required to accept it. For example, you might write the content of the prompt to ensure users do not access data unnecessarily or search for information about people they know that are not under investigation.

Procedure

Complete the following steps to enable and configure the privacy prompt:

1. Modify the `toolkit/configuration/fragments/common/privacyagreement.html` file to display the content of your privacy prompt.
2. Configure i2 Analyze to enable the privacy prompt and specify how frequently users must accept it.
   a. To enable the prompt, in the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file, set the `EnablePrivacyPrompt` property to `true`. For example:

      ```
      EnablePrivacyPrompt=true
      ```

   b. To configure the time period before users receive the prompt again, in the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file, set the `PrivacyAcceptancePeriodDays` property to the value in days when the prompt should be displayed again. For example:

      ```
      PrivacyAcceptancePeriodDays=90
      ```

      If you do not set a value for `PrivacyAcceptancePeriodDays`, users will not receive another prompt until all privacy agreements are reset.
3. Run the following command from the `toolkit/scripts` directory on your Liberty server to update the application with your changes:

   ```
   setup -t deployLiberty
   ```

Results

The next time that a user logs in to i2 Analyze, they must accept or reject your privacy prompt.

What to do next

You can reset the privacy prompt acceptance status for all users by using a deployment toolkit task. For example, if you change the wording of the prompt, you might want to ensure that all analysts accept the new agreement. To reset the acceptance status, run the following commands from the `toolkit/scripts` directory on your Liberty server:

```
setup -t resetPrivacyAgreements
setup -t restartLiberty
```

You can also complete these actions on a running server by using the associated admin endpoints. Configuration changes completed by the admin endpoints do not persist after a server redeployment. For more information about the admin endpoints, see Using the admin endpoints.

Setting a deployment name

If your i2 Analyze deployment is one of several that your users might connect to, it's convenient for them to see the deployment name in a prominent location. You can provide a display name for your deployment through a setting in the `DiscoServerSettingsCommon.properties` file.

About this task

To set the deployment name, you configure it in the server settings and then redeploy the server.

Procedure

Complete the following steps to provide a display name for your i2 Analyze deployment:

1. Come up with a name for your deployment that distinguishes it from others that your users might connect to.

   i2 Analyze places no hard restrictions on the length of the name or the characters it uses, but shorter is generally better for client user interfaces.
2. Configure i2 Analyze to use the name. In the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file, set the `DeploymentDisplayName` property:

   ```
   DeploymentDisplayName=<your deployment name>
   ```
3. Run the following commands from the `toolkit/scripts` directory on your Liberty server to update the application with your changes:

   ```
   setup -t stopLiberty
   setup -t deployLiberty
   setup -t startLiberty
   ```

Results

The next time that a user logs in to i2 Analyze from the Analyst's Notebook desktop client or the i2 Notebook web client, they see the deployment name in the user interface.

- Analyst's Notebook displays the deployment name at the top of the application window, next to the user name.
- i2 Notebook displays the deployment name in the User menu, beneath the user name.

Configuring server monitoring

You can monitor and report on your deployment of i2 Analyze by using Prometheus and Grafana.

The monitoring of i2 Analyze licences is completed via a REST API. By using a REST API, i2 Analyze can be monitored by well-used industry standard tools.

The i2 Analyze toolkit includes template configuration files for Prometheus and Grafana. The instructions here describe how you can install and configure Prometheus and Grafana to monitor your deployment of i2 Analyze.

Running Prometheus and Grafana

i2 Analyze and its components provide a number of metrics through REST endpoints. You can use Prometheus to monitor your deployment, and Grafana to visualize the captured data.

About this task

To monitor your deployment of i2 Analyze, complete the following actions:

- Update the i2 Analyze configuration to enable metrics reporting
- Configure and run Prometheus and Grafana
- View the monitoring reports in Grafana using the provided dashboards

To configure your Prometheus and Grafana instances, use your configuration development environment. When the tools are configured to meet your requirements, you can update them to match the security requirements of your production system.

Complete the following steps to enable metrics on i2 Analyze, and configure and install the monitoring tools.

Procedure

1. Configure i2 Analyze.
   a. Uncomment the monitoring features and non-SSL elements in the Liberty `server.extensions.xml` file. The file is located in the `toolkit/configuration/liberty` directory.

      For example:

      ```
      <server>
          <featureManager>
              <feature>mpMetrics-1.1</feature>
              <feature>monitor-1.0</feature>
          </featureManager>

          <mpMetrics authentication="false"/>
      </server>
      ```

   b. To enable i2 Analyze metrics, in the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file, set the `EnableMetrics` property to `true`.

      For example:

      ```
      EnableMetrics=true
      ```

   c. Run the following command on your Liberty server to update the application with your changes.

      ```
      setup -t deployLiberty
      ```
2. Configure and run Prometheus.

   When you start Prometheus, you must reference a configuration file. The i2 Analyze deployment toolkit includes a template Prometheus configuration file that contains configuration to enable monitoring of Liberty and i2 Analyze. The `prometheus.yml` file is in the `toolkit/examples/prometheus` directory.

   In the `prometheus.yml` file, update the values to match your environment:

   a. Replace `<liberty_domain>:<port>` with the fully qualified domain name and port of your Liberty server.

For example, `i2analyze.eia:9082`

Note: When Liberty is not running in Docker and Prometheus is running in a Docker container hosted on the same machine, use the following value for `<liberty_domain>`: `host.docker.internal`.

If you are using the opal-daod deployment pattern, update the `metrics_path` to `/opaldaod/api/v1/metrics`.

After you complete the configuration file, install and run Prometheus with your configuration. In this example we install and run Prometheus using Docker, but you can install it in a number of ways. For more information about installing Prometheus, see Prometheus installation.

a. If you have Docker installed, run the following command to run Prometheus with your configuration on Docker:

```
docker run -d
    -p 9090:9090
    --network=<network_name>
    -v "/<path_to>/prometheus.yml:/etc/prometheus/prometheus.yml"
    prom/prometheus
```

Where:

- `<network_name>` is the name of a Docker network for the container to run on. For more information about creating a Docker network, see Docker network.
- `<path_to>` is the path with your updated `prometheus.yml` file. For example, `toolkit/examples/prometheus`.

3. Configure and run Grafana.

The i2 Analyze deployment toolkit includes a template Grafana data source configuration file and dashboards. The `prometheus-datasource.yml` file is in the `toolkit/examples/grafana/provisioning/datasources` directory.

a. In the `prometheus-datasource.yml`, replace `<prometheus_domain>:<port>` with the host name and port of your Prometheus server.

The quickest way to install Grafana is by using Docker, but you can install it in a number of other ways. For more information about installing Grafana, see Install Grafana.

b. If you have Docker installed, run the following command to run Grafana on Docker:

```
docker run -d
    --name "grafana"
    -p "3000:3000"
    --network=<network_name>
    -v "<grafana_data>:/var/lib/grafana"
    -v "<grafana_provisioning_dashboards>:/etc/grafana/provisioning/
dashboards"
    -v "<grafana_provisioning_datasources>:/etc/grafana/provisioning/
datasources"
    -v "<grafana_dashboards>:/etc/grafana/dashboards"
    "grafana/grafana-oss"
```

Where:

- `<network_name>` is the name of a Docker network for the container to run on. Prometheus and Grafana must be able to communicate with each other. When both are running in Docker, ensure they are running on the same Docker network. For more information about creating a Docker network, see Docker network.

- `<grafana_data>` is a path to store Grafana data.
- `<grafana_provisioning_dashboards>` is the path to your dashboard provisioning files. For example `toolkit/examples/grafana/provisioning/dashboards`. For more information, see Provisioning dashboards.
- `<grafana_provisioning_datasources>` is the path to your data source configuration files. For example `toolkit/examples/grafana/provisioning/datasources`. For more information, see Provisioning data sources".
- `<grafana_dashboards>` is the path to your dashboard files. For example `toolkit/examples/grafana/dashboards/provisioning`.

You can import your own dashboards into Grafana, to do this you must start Grafana with a different command. For more information, see Importing Grafana dashboards

What to do next

Connect to the Grafana UI to review the dashboards. In a browser, navigate to your Grafana instance. For example, `http://localhost:3000/dashboards`.

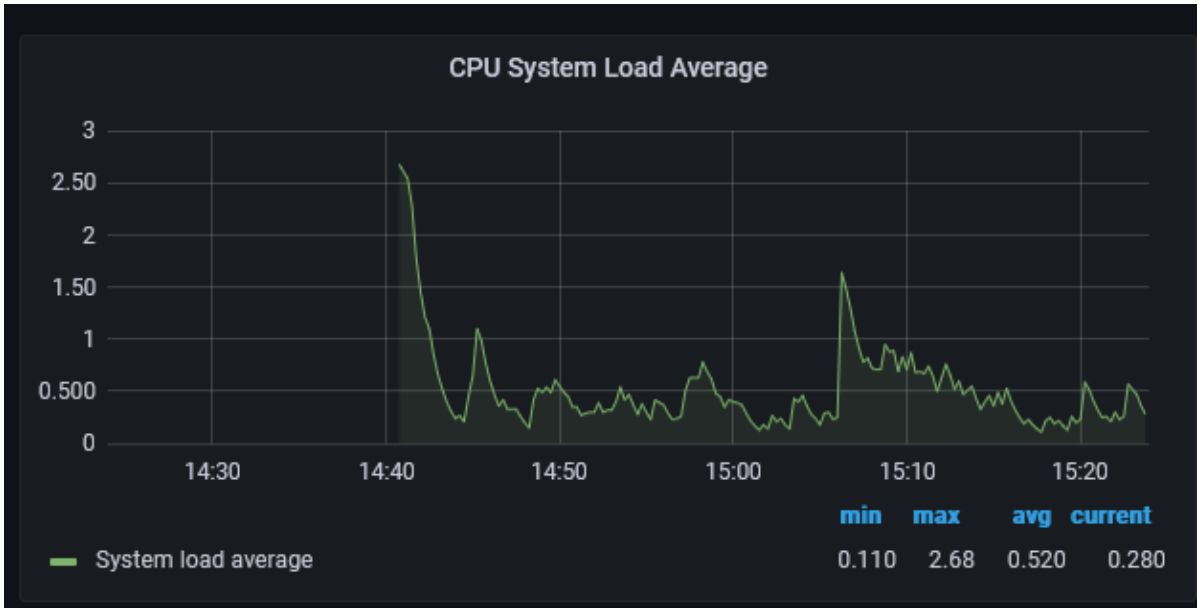
By default, Grafana includes an admin user that you can use to log in. The user name is "admin" and the password is "admin".

If you have chosen to manually import your dashboards, complete the steps in Importing Grafana dashboards to see the example dashboards.

The toolkit includes the following example dashboards:

- The "i2 Analyze License Reporting" dashboard reports on the usage of your system by analysts. For example: User Activity, Privacy Agreement Count, and License Consumption. For example:

- The "Liberty-Metrics" dashboard reports on the hardware usage of the Liberty server. For example: CPU Load, Heap Usage, and JVM Uptime. For example:



You can extend the example configuration to meet your requirements.

To run Prometheus and Grafana in production, you might want to configure authentication, authorization, and other features. You can use the Prometheus and Grafana documentation to determine how to configure these features.

- When you move into pre-prod and production deployment, you might be deploying with TLS. You can configure Prometheus to communicate with Liberty via TLS by modifying the `prometheus.yml` file:
  - <tls_config>
  - To configure TLS, you must provide a CA certificate that trusts the certificate received from the Liberty server.
  - You must also update the `scheme` value for the Liberty job to use the value `https`
- You can configure user authentication:
  - Grafana: Configure authentication
  - Prometheus: HTTPS and authentication

For the complete documentation, see:

- Prometheus
- Grafana

Importing Grafana dashboards

If you would rather import dashboards than have them already set up when you start Grafana, start your container without the dashboard and datasource volumes.

To start the Grafana container without the dashboard and datasource volumes, run:

```
docker run -d
```

```
--name "grafana"
-p "3000:3000"
--network=<network_name>
-v "<grafana_data>:/var/lib/grafana"
"grafana/grafana-oss"
```

After Grafana is started, connect to the Grafana UI and log in By default, Grafana includes an admin user that you can use to log in. The user name is "admin" and the password is "admin". Navigate to **Import dashboard -> Dashboards -> Grafana** and import your dashboard. Some example dashboards are provided in the `toolkit/examples/grafana/dashboards/import-configurations` directory.

After a dashboard is imported, you must create a `datasource` for your dashboard to connect to. The example `import-configurations` require a Prometheus datasource with the name "prometheus". Navigate to **Add data source -> Data sources -> Connections -> Grafana** and select the **Prometheus** datasource. Specify the name "prometheus" and the connection url for your Prometheus instance. If you are using the container example from earlier, the URL will be similar to: `http://prometheus.eia:9090`.

The default settings can be left unchanged. Navigate to the bottom of the page and click **Save & Test**. The UI displays that the source has been connected to successfully.

Configuring system limits and variables

The i2 Analyze deployment toolkit contains settings that specify system limits and variables. If you modify these settings in a deployed system, redeploy i2 Analyze to update your deployment with any changes.

Modifying the database configuration

You can configure the Information Store database that is deployed as part of i2 Analyze. The types of configuration that you can modify include certain properties of the database and the indexes that are created.

Information Store database properties

The Information Store database properties files specify the names and details of database objects that are used in the Information Store. There are separate files for IBM® Db2®, Microsoft® SQL Server, and PostgreSQL.

The `InfoStoreNamesDb2.properties`, `InfoStoreNamesSQLServer.properties`, and `InfoStoreNamesPostgres.properties` files are in the `toolkit\configuration\fragments\opal-services-is\WEB-INF\classes` directory.

The settings in these files control the following database objects:

- The names of the database schemas
- The name of the deletion by rule role
- The collation sequence
- For Db2, the page size
- For Db2, the names of the buffer pools
- For SQL Server, the logical names and locations of the filegroup files

Note: If you change the values for any of the settings after the Information Store is created, you must either manually change the names of the database objects in your database management system or re-create the database.

Settings for Db2, SQL Server, and PostgreSQL

The `InfoStoreNamesDb2.properties`, `InfoStoreNamesSQLServer.properties`, and `InfoStoreNamesPostgres.properties` files contain the following settings:

`MetadataSchema`

The metadata schema contains objects that relate to the definitions of data structures in the Information Store.

By default, the value is `IS_Meta`

`DataSchema`

The data schema contains objects that store all the data that is available for analysis.

By default, the value is `IS_Data`

`WebChartSchema`

The web chart schema contains temporary objects used during manipulation of the web chart.

By default, the value is IS_WC

`VisualQuerySchema`

The visual query schema contains temporary objects used during visual query processing.

By default, the value is `IS_Vq`

`FindPathSchema`

The find path schema contains objects that support find path results.

By default, the value is `IS_FP`

`StagingSchema`

The staging schema contains temporary objects that support the ingestion process.

By default, the value is `IS_Staging`

`PublicSchema`

The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.

By default, the value is `IS_Public`

`DeletionByRuleRoleName`

The deletion-by-rule role name.

By default, the value is `Deletion_By_Rule`

`Collation`

The collation sequence used for the Information Store. You can only change this setting before you create the Information Store database. Defaults to `CLDR181_LEN_S1` for Db2 and `Latin1_General_100_CI_AI_SC` for SQL Server.

Note: For PostgreSQL, collation works a little differently and there are three settings instead of one. For more information, see The `InfoStoreNamesPostgres.properties` file.

The `InfoStoreNamesDb2.properties` file

The following properties are in the `InfoStoreNamesDb2.properties` file:

Property	Description	Default value
MetadataSchema	The metadata schema contains objects that relate to the definitions of data structures in the Information Store.	`IS_Meta`
DataSchema	The data schema contains objects that store all the data that is available for analysis.	`IS_Data`
WebChartSchema	The web chart schema contains temporary objects used during manipulation of the web chart.	`IS_WC`
VisualQuerySchema	The visual query schema contains temporary objects used during visual query processing.	`IS_Vq`
FindPathSchema	The find path schema contains objects that support find path results.	`IS_FP`
StagingSchema	The staging schema contains temporary objects that support the ingestion process.	`IS_Staging`
PublicSchema	The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.	`IS_Public`
SystemTempTableSpace	The system temporary table space.	`IS_TEMP_TS`
UserTemp16KTableSpace	The user temporary 16K table space to hold global temporary objects.	`IS_16K_TS`
BigTableSpace	The partitioned table space to hold data objects.	`IS_BIG_TS`

Property	Description	Default value
BigIndexTableSpace	The partitioned table space to hold data indexes.	`IS_BIG_INDEX_TS`
SmallTableSpace	The non-partitioned table space to hold data objects.	`IS_SMALL_TS`
SmallIndexTableSpace	The non-partitioned table space to hold data indexes.	`IS_SMALL_INDEX_TS`
LobTableSpace	The table space to hold LOB data objects.	`IS_LOB_TS`
LobIndexTableSpace	The table space to hold LOB data indexes.	`IS_LOB_INDEX_TS`
InternalStagingTableSpace	The table space to hold internal staging data objects created during ingestion Note: these tables have data inserted using NOT LOGGED INITIALLY	`IS_INTSTG_TS`
PageSize	The page size.	`16K`
BufferPool16K	The 16K buffer pool.	`IS_16K_BP`
DeletionByRuleRoleName	The deletion-by-rule role name.	`Deletion_By_Rule`
Collation	The collation sequence used for the Information Store in Db2. You can only change this setting before you create the Information Store database. Defaults to "CLDR181_LEN_S1".	`CLDR181_LEN_S1`

The `InfoStoreNamesSQLServer.properties` file

The following properties are in the `InfoStoreNamesSQLServer.properties` file:

Property	Description	Default value
ISSchema	The Information Store schema contains internal configuration information about the Information Store database.	`IS_Core`
MetadataSchema	The metadata schema contains objects that relate to the	`IS_Meta`

Property	Description	Default value
	definitions of data structures in the Information Store.	
DataSchema	The data schema contains objects that store all the data that is available for analysis.	`IS_Data`
WebChartSchema	The web chart schema contains temporary objects used during manipulation of the web chart.	`IS_WC`
VisualQuerySchema	The visual query schema contains temporary objects used during visual query processing.	`IS_Vq`
FindPathSchema	The find path schema contains objects that support find path results.	`IS_FP`
StagingSchema	The staging schema contains temporary objects that support the ingestion process.	`IS_Staging`
PublicSchema	The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.	`IS_Public`
DeletionByRuleRoleName	The deletion-by-rule role name.	`Deletion_By_Rule`
Collation	The collation sequence used for the Information Store in SQL Server. You can only change this setting before you create the Information Store database. Defaults to Latin1_General_100_CI_AI_SC.	`Latin1_General_100_CI_AI_SC`
PrimaryName	The SQL Server database primary data file contains the startup and configuration information for the Information Store database. The logical name of the primary data file.	`IStore_System_Data`

Property	Description	Default value
PrimaryFilename	The filename or absolute path for the primary data file.	`IStore-p1.mdf`
UserTableName	The user table filegroup contains all the tables and data for the Information Store database. The logical name of the user table in the user table filegroup.	`User_Table_Data`
UserTableFilename	The filename or absolute path for the user table file.	`IStore-ut1.ndf`
IndexName	The index filegroup contains the indexes for the Information Store database. The logical name of the index file.	`IStore_Indexes`
IndexFilename	The filename or absolute path for the index file.	`IStore-i1.ndf`
LobName	The LOB filegroup contains the tables that Store LOB data. The logical name of the lob file.	`IStore_Lobs`
LobFilename	The filename or absolute path for the index file.	`IStore-b1.ndf`
MemoryOptimizedName	The memory-optimized filegroup holds one or more containers that contain data files, delta files, or both for memory-optimized tables. The logical name of the memory-optimized file.	`IStore_Memory_Optimized`
MemoryOptimizedFilename	The filename or absolute path for the memory-optimized file.	`IStore-mo1.ndf`
SqlLogName1	The transaction log files used for the Information Store. The logical name of the first SQL log.	`IStore_Logs_1`
SqlLogFilename1	The filename or absolute path for the first SQL log file.	`IStore-log1.ldf`
SqlLogName2	The logical name of the second SQL log.	`IStore_Logs_2`

Property	Description	Default value
SqlLogFilename2	The filename or absolute path for the second SQL log file.	`IStore-log2.ldf`

The `InfoStoreNamesPostgres.properties` file

The following properties are in the `InfoStoreNamesPostgres.properties` file:

Property	Description	Default value
ISSchema	The Information Store schema contains internal configuration information about the Information Store database.	`IS_Core`
MetadataSchema	The metadata schema contains objects that relate to the definitions of data structures in the Information Store.	`IS_Meta`
DataSchema	The data schema contains objects that store all the data that is available for analysis.	`IS_Data`
WebChartSchema	The web chart schema contains temporary objects used during manipulation of the web chart.	`IS_WC`
VisualQuerySchema	The visual query schema contains temporary objects used during visual query processing.	`IS_Vq`
FindPathSchema	The find path schema contains objects that support find path results.	`IS_FP`
StagingSchema	The staging schema contains temporary objects that support the ingestion process.	`IS_Staging`
PublicSchema	The public schema contains objects that represent a public API for the Information Store. It also contains procedures, tables, and views related to the deletion by rule feature.	`IS_Public`
DeletionByRuleRoleName	The deletion-by-rule role name.	`Deletion_By_Rule`

Property	**Description**	**Default value**
Collation_Locale	The (ICU) collation used for the Information Store in PostgreSQL. You can only change these settings before you create the Information Store database. If Collation_Deterministic is true, all operators are case- and accent-sensitive If Collation_Deterministic is false, "exact" operators follow the rules of the Collation_Locale the "like" operator is governed by Collation_Sensitivity Defaults to: en-US-u-ks-level1 and false (US English, case- and accent-insensitive)	`en-US-u-ks-level1`
Collation_Deterministic		`false`
Collation_Sensitivity	The case- and accent-sensitivity of the "like" operator when Collation_Deterministic is false Must be CIAI, CIAS, CSAI, or CSAS Defaults to: CIAI (case- and accent-insensitive)	`CIAI`
CollateClause	The collation used throughout columns in the Information Store in PostgreSQL. This setting defaults to 'COLLATE public.istore_collation'. It is required for correct sorting of text based columns when using a non-deterministic collation. If you plan on using a deterministic collation (such as a case-sensitive collation) then set this property to be empty.	`COLLATE public.istore_collation`
DatabaseTablespace	The default tablespace for the Information Store database.	`pg_default`
BigDataTablespace	The tablespace for big data tables.	`pg_default`
BigIndexTablespace	The tablespace for indexes on big data tables.	`pg_default`

Property	Description	Default value
SmallDataTablespace	The tablespace for small data tables.	`pg_default`
SmallIndexTablespace	The tablespace for indexes on small data tables.	`pg_default`

Creating indexes in the Information Store database

In the Information Store, you can add indexes to the item type tables, and to the property type columns of those tables.

About this task

Indexes can improve the performance of Visual Query searches. Indexes can also improve the performance of the merged property values definition views if you are using them.

To create an index, you must use the `informationStoreModifications.sql` file. This script is run every time the Information Store database is created. You can also use the `modifyInformationStoreDatabase` toolkit task to run the script at any time.

The file must be in the `configuration\environment\opal-server\databases\<infostore>` directory. `infostore` is the value of the `id` attribute of the `database` element for your Information Store database in the `topology.xml` file.

In the Information Store database, the `IS_DATA` schema contains the tables for each item type. Entity item type tables are prefixed with `E_` and link item type tables with `L_`. In these tables, property columns are prefixed with `P_`.

For example, your i2 Analyze schema might contain a Person entity with a First Given Name property. If you know that this property is used a lot by analysts in Visual Query searches, you might want to create an index on that table and column. To create a simple index for the First Given Name property, add the following statement to the `informationStoreModifications.sql` file:

```
CREATE INDEX E_PERSON_FN_IX ON IS_DATA.E_PERSON (P_FIRST_GIVEN_NAME);
```

`E_PERSON_FN_IX` is the name of the index to create, `IS_DATA.E_PERSON` is the table for the Person entity type, and `P_FIRST_GIVEN_NAME` is the column for the first given name property type.

To determine the syntax of the SQL statement that you must use to create the index, use the documentation for your database management system. For more information about creating indexes in a Db2 database, see CREATE INDEX statement; for SQL Server, see CREATE INDEX (Transact-SQL); and for PostgreSQL see CREATE INDEX.

Procedure

1. Identify the item types, and any of their property types, that you want to add indexes for.
2. Create the directory for the `informationStoreModifications.sql` file.
   a. In the `configuration\environment\opal-server` directory, create the `databases` directory.
   b. In the `databases` directory that you created, create the *`infostore`* directory. You can find the value to use for *`infostore`* in your `topology.xml` file. The value to use in your deployment is

the value of the `id` attribute of the `<database>` element for your Information Store database. For example, `configuration\environment\opal-server\databases\infostore`.

3. Using a text editor, create a file that is named `informationStoreModifications.sql` in the `configuration\environment\opal-server\databases\infostore` directory.
4. Develop a script to create the indexes on the tables and columns that you identified in step 1 in the `informationStoreModifications.sql` file.
5. Save and close the file.
6. If the Information Store database exists, run the `modifyInformationStoreDatabase` toolkit task to run the script that you saved in step 4.
   a. Open a command prompt and navigate to the `toolkit\scripts` directory.
   b. Run the following command to run the `informationStoreModifications.sql` file:

      ```
      setup -t modifyInformationStoreDatabase
      ```

What to do next

Ensure that the indexes you expect are in the Information Store database. You can use IBM Data Studio or SQL Server Management Studio to see the indexes in the Information Store database.

Modifying the remote Db2 database configuration

You can change the configuration attributes of your remote Db2 databases in the i2 Analyze deployment toolkit. To modify the remote database configuration, you must recatalog the remote nodes.

About this task

When you recatalog a remote node, the existing node is removed from the Db2 node directory and is then cataloged again with the updated configuration values.

If you need to configure security settings on the connection, you can recatalog the remote nodes manually, instead of using the tasks that are in the deployment toolkit. To recatalog the remote nodes manually, you can use the Db2 `UNCATALOG NODE` and `CATALOG TCPIP/TCPIP4/TCPIP6 NODE` commands.

You run the `UNCATALOG NODE` and `CATALOG NODE` commands manually instead of using the `recatalogRemoteDB2Nodes` task. For more information about the commands, see `UNCATALOG NODE` command and `CATALOG TCPIP/TCPIP4/TCPIP6 NODE` command.

Note: The `catalogRemoteDB2Nodes` and `recatalogRemoteDB2Nodes` tasks use the Db2 `CATALOG TCPIP NODE` command. The following table shows how the `CATALOG` command parameters map to the values in the `topology.xml` file:

`CATALOG TCPIP NODE` command parameters	**`<database>` element attributes**
`TCPIP NODE` *`nodename`*	`node-name`
`REMOTE` *`hostname`*	`host-name`
`SERVER` *`port number`*	`port-number`
`REMOTE_INSTANCE` *`instance-name`*	`instance-name`

Procedure

1. Change the host name, port, and operating system type attributes of the remote database:

a. Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
b. Update the `host-name` and `port-number` attribute values of the `<database>` element for the database.
c. Run the following command to recatalog the remote node:

```
setup -t recatalogRemoteDB2Nodes
```

2. Change the node name attribute of the remote database:
   a. Run the following command to uncatalog the remote node:

```
setup -t uncatalogRemoteDB2Nodes
```

   b. Using an XML editor, open `toolkit\configuration\environment\topology.xml`.
   c. Update the `node-name` attribute values of the `<database>` element for the database.
   d. Run the following command to catalog the remote node:

```
setup -t catalogRemoteDB2Nodes
```

What to do next

The next time that you deploy i2 Analyze, or recreate the database, the database is created using the new values that you provided in the `topology.xml`.

Modifying the maximum number of notes in a record

In a deployment of i2 Analyze, analysts can add notes that contain information that is not categorized by the type of entity or link to Information Store records. The number of notes that a record contains can impact the performance of searching for and uploading that record.

About this task

You can configure the maximum number of notes that a record can contain to a number that better matches system requirements. By default, the maximum number of notes that a record can contain is 50.

If a record already contains more notes than the maximum that you set, analysts can continue to edit and delete the existing notes. However, analysts can only create notes in the record by deleting existing notes until the number of notes is less than the new maximum.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Change the value of the `RecordMaxNotes` property. For example, `RecordMaxNotes=25`.
3. Save your changes.

What to do next

After you modify the maximum number of notes a record can contain, redeploy the i2 Analyze application for the changes to take effect. For more information about redeploying your system, see Redeploying Liberty.

After you redeploy i2 Analyze, test that the system continues to meet your requirements.

Configuring search

Depending on the requirements of the environment that you are deploying into, a number of configurable options are available for searching and indexing. You can configure the behavior of search features to match your requirements.

Configuring Quick Search

Analysts can use Quick Search to find records in the Information Store by specifying terms that records might contain. You can configure the behavior of Quick Search to meet the language and structure of your data and the requirements of your analysts.

Configuring the Solr index

By default, the Solr index assumes that text data in i2 Analyze follows the conventions of Western European languages, and uses a list of synonyms that contains US English terms. It also treats letters like o, ó, and ö as equivalent for the purposes of filtering search results. You can change these settings so that the index meets the language requirements of your data.

About this task

To configure the Solr index, modify the `schema-template-definition.xml` file. In the template definition, you can control the following aspects of Solr:

- How Solr returns data based on the language of the data in the index
- The Solr synonyms file
- Whether to preserve diacritics in search result filters. For example to treat a, å, and ä as equivalent or not.

By default, the Solr search index is configured for Western European languages, and uses synonym lists that contain US English terms. The template definition configuration file includes template options for `ar_EG` (Arabic) and `he_IL` (Hebrew) languages.

Each language template has a default synonyms file that is associated with the template language. You can change the default synonyms file to use a customized synonyms file. For more information, see Creating a synonyms file.

By default, diacritics are not preserved in filters. This means that "Amélie" and "Amelie" both contribute to the "amelie" filter To ensure that "Amélie" and "Amelie" create separate filters, you can preserve the diacritics.

Note: Changing the diacritic behavior for filters does not affect the behavior when searching.

If you need to configure the behavior of the Solr index, do this in your configuration development environment. If your system contains data, you must reindex after changing the Solr configuration.

Procedure

1. To modify the Solr index configuration, open the `configuration\solr\schema-template-definition.xml` file in an XML editor.
2. Specify the language template and synonyms file for the index.
   a. To specify the language template and default synonyms file, uncomment the section of the template file for the language you want to use.

For example, to use Arabic uncomment the `<Definition>` and `<SynonymsFile>` file elements in the `ar_EG config` section:

```
<!-- ar_EG config -->
  <Definition Analyzer="free_text">
    <AnalyzerChain>
      <tokenizer class="solr.StandardTokenizerFactory"/>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ASCIIFoldingFilterFactory"
 preserveOriginal="false"/>
    </AnalyzerChain>
    <PostSynonyms>
      <filter class="solr.ArabicNormalizationFilterFactory"/>
      <filter class="solr.ArabicStemFilterFactory"/>
    </PostSynonyms>
  </Definition>

  <Definition Analyzer="text_facet">
    <AnalyzerChain>
      <filter class="solr.LowerCaseFilterFactory"/>
      <filter class="solr.ASCIIFoldingFilterFactory"
 preserveOriginal="false"/>
      <filter class="solr.ArabicNormalizationFilterFactory"/>
      <filter class="solr.ArabicStemFilterFactory"/>
    </AnalyzerChain>
  </Definition>

  <SynonymsFile Path="synonyms-ar_EG.txt" />
```

b. To specify a different synonyms file, place the file in the `configuration\solr` directory and provide the file name in the `<SynonymsFile>` element.

For example:

```
<SynonymsFile Path="custom-synonyms.txt" />
```

For more information about creating a custom synonyms file, see Creating a synonyms file.

3. To preserve the diacritics in facets, you must complete the previous step for your chosen template and remove the following line from the `text_facet` analyzer:

```
<filter class="solr.ASCIIFoldingFilterFactory" preserveOriginal="false"/>
```

Update the application with your configuration changes.

Note: This procedure removes the data from your Solr index. When Liberty is started, Solr reindexes the data in the Information Store database.

1. On the Liberty server, open a command prompt and go to the `toolkit\scripts` directory.

2. Stop i2 Analyze.

a. If you are using a single server deployment, run `setup -t stop`.

b. If you are using a multiple server deployment, complete the steps to stop the components of i2 Analyze in Stopping and starting i2 Analyze.

3. Redeploy i2 Analyze:

```
setup -t deployLiberty
```

4. Create and upload the Solr configuration to ZooKeeper:

```
setup -t createAndUploadSolrConfig --hostname 'liberty.host-name'
```

Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

5. Clear the search index:

   ```
   setup -t clearSearchIndex --hostname 'liberty.host-name'
   ```

   Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

6. Start i2 Analyze

   a. If you are using a single server deployment, run `setup -t start`.

   b. If you are using a multiple server deployment, complete the steps to start the components of i2 Analyze in Stopping and starting i2 Analyze.

What to do next

Run a selection of queries against your deployment server to test the configuration.

The Solr template configuration file

The Solr template configuration file that is provided in the i2 Analyze deployment toolkit contains commented-out templates for a number of languages.

Root element

- `<SolrSchemaTemplate>`

i2 Analyze elements

- `<Definition>`
- `<AnalyzerChain>`
- `<PostSynonyms>`
- `<SynonymsFile>`

Solr elements

- `<tokenizer>` and `<filter>`

Root element

`<SolrSchemaTemplate>`

`<SolrSchemaTemplate>` is the root element of the Solr template configuration file. It references the `SolrSchemaTemplate.xsd` file and version number. Do not change the value for the `Version` attribute.

```
<SolrSchemaTemplate
  xmlns:tns="http://www.i2group.com/Schemas/2021-02-12/SolrSchemaTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.i2group.com/Schemas/2021-02-12/
SolrSchemaTemplate SolrSchemaTemplate.xsd "
  Version="1">

</SolrSchemaTemplate>
```

i2 Analyze elements

<Definition>

In the Solr index, analyzers are used to examine the text that is stored in the index. Two of the analyzers that i2 Analyze uses are the `free_text` and `text_facet` analyzers. In the template file, the analyzer is specified in the `Analyzer` attribute of the `<Definition>` element.

```
<Definition Analyzer="free_text">
...
</Definition>

<Definition Analyzer="text_facet">
...
</Definition>
```

<AnalyzerChain>

The `<AnalyzerChain>` element is a container for the Solr elements in the analyzer chain. The `<AnalyzerChain>` element is a child of the `<Definition>` element.

```
<Definition Analyzer="free_text">
  <AnalyzerChain>
  ...
  </AnalyzerChain>
</Definition>
```

<PostSynonyms>

The `<PostSynonyms>` element is a container for the Solr elements in the analyzer chain that are applied after the synonym filter. The `<PostSynonyms>` element is a child of the `<Definition>` element, specified after the `<AnalyzerChain>` element.

```
<Definition Analyzer="free_text">
  <AnalyzerChain> ... </AnalyzerChain>
  <PostSynonyms>
  ...
  </PostSynonyms>
</Definition>
```

<SynonymsFile>

The `Path` attribute of the `<SynonymsFile>` element contains the file name of the synonyms file to use. The `<SynonymsFile>` element is at the same level as the `<Definition>` element.

```
<SynonymsFile Path="synonyms-ar_EG.txt" />
```

Solr elements

The `<tokenizer>` and `<filter>` are directly converted to Solr. For more information about the elements, see Understanding Analyzers, Tokenizers, and Filters.

In the Solr template configuration, the `<tokenizer>` and `<filter>` elements can be child elements of the `<AnalyzerChain>` and `<PostSynonyms>` elements.

Note: The `<tokenizer>` element can be specified as a child of the `<Definition>` element where `Analyzer="free_text"`.

Creating a synonyms file

Solr provides the facility to configure the synonyms that are used for querying textual data. In i2 Analyze, you can use this option to apply a customized list of synonyms at query time. Synonyms, if not

accounted for, can cause a reduction in the relevance of a search result when you search for keywords that are present in alternative forms in your index.

About this task

The synonyms file is the part of the Solr configuration that accounts for the presence of synonyms in your data. For example, your data might contain the words, "bag, handbag, pocketbook, purse" for the concept "bag". When someone searches they are likely to search for one, but expect results for all four.

To meet that expectation, you might want to create a customized synonyms file to accommodate similar variations that are specific to your data. The exact words in a synonyms list that are most useful in your deployment depend on the content of your data. You can also use a mix of languages, which might be useful in some contexts, for example names: 'George, ########, Jorge'.

The default synonyms file and synonyms list are in US English. The synonyms files that are associated by default with each supported language are supplied in the directory, `toolkit\configuration\solr`.

To customize the alternative terms that are used in search operations for your data, you can create files that contain different terms from those terms that are contained in the supplied synonyms files.

The customized file must adhere to the following guidelines:

- The file must be UTF-8 encoded.
- The terms in the file must match the terms that are produced by the analyzer chain that is used in Solr prior to the synonym filter being applied.
- If multiple forms of a word exist, all the forms must be specified in order for synonym matching to work on each form.
- Words from Latin script languages, for example French or Italian, must be specified without diacritics. For example, use the following substitution:
  - a instead of á
  - c instead of ç
- Arabic and Hebrew words must be specified exactly as they are written.

Procedure

1. Create a text file that defines synonyms in the required Solr format.

   For more information, see https://lucene.apache.org/core/8_2_0/analyzers-common/org/apache/lucene/analysis/synonym/SolrSynonymParser.html.

   Notes:

   a. You cannot search for multi-word terms. However, if you have data that contains terms "USA" and "United States of America", you can search for "USA" and use a synonym to ensure a match with "United States of America".

   b. You can provide synonyms for terms that include punctuation. However, a search on such a term might not work correctly. The unexpected result is because a filter is applied before synonyms, which means, for example, "Mary-Ann" becomes "Mary,Ann" and then synonyms are expanded from "Mary and "Ann"; not "Mary-Ann" or "Maryann".
2. Save the file with a `.txt` extension, for example `custom-synonyms.txt`.
3. Complete the instructions in Configuring the Solr index to deploy with your synonyms file.

Modifying the wildcard minimum character limits

Wildcard characters are used to match zero or more alphanumeric characters in a search term. You can configure how users complete Quick Search and Visual Queries by modifying the minimum number of characters that must be included in a search term that includes wildcards.

About this task

The following wildcard characters are available in i2 Analyze:

- *****

  Matches zero or more alphanumeric characters in this position.

  For example, the search term `Tim*` matches `Tim`, `Time`, and `Timely`.

- **?**

  Matches one alphanumeric character in this position.

  For example, the search term `Tim?` matches `Time`.

Wildcard characters can be used at any position in a search term.

Search terms with wildcard queries might result in a large number of matches, which might cause performance problems. To reduce the possible matches from a wildcard search, you can ensure that users provide a minimum number of characters with a wildcard. For example, the term "`*`" matches everything. If users must provide a minimum of 3 characters with an asterisk, for example the term "`abc*`", the number of matches is reduced to values that begin with "abc".

If the minimum number of characters is set too high, users might not be able to search for the terms that they need. For example, in a deployment where the wildcard minimum characters are configured as follows:

- A minimum of 3 characters other than asterisks (*) must be provided in a term.
- A minimum of 2 characters other than question marks (?) and asterisks (*) must be provided in a term.

If the user knows only 2 characters of a license plate, they might not be able to use a wildcard search:

- If the user knows the position of the 2 characters, they can use the question mark wildcard character in the search. If the 2 characters are in positions 2 and 3, then the query "`?AB*`" is valid in the configuration that is described.
- If the position of the characters is not known, the user might want to search for "`*AB*`", which is invalid in the configuration that is described.

In addition to wildcard characters that are specifically entered as part of Visual Query, several conditions provide implicit wildcard logic:

- 'Starts with' - Applies an asterisk to the end of the condition. For example, 'Starts with: Fred' is equivalent to 'Fred*', which could match Fred, Frederick, or Freddie.
- 'Ends with' - Applies an asterisk to the start of the condition. For example, 'Ends with: Fred' is equivalent to '*Fred', which could match Fred, Wilfred, or Alfred.
- 'Contains' - Applies an asterisk to both the start and the end of the condition. For example, 'Contains: Fred' is equivalent to '*Fred*', which could match any of the above terms, but also include Alfredo.

The use of these conditions follow the same limits as wildcard characters that have been entered explicitly.

To change the minimum number of characters that must be included in a search query with a wildcard character, edit properties in `DiscoServerSettingsCommon.properties`.

The properties that specify the minimum number of characters for Quick Search are:

- **`WildcardMinCharsWithAsterisk`**

  The minimum number of characters other than asterisks (*) that must be included in a wildcard query that contains an asterisk.

- **`WildcardMinCharsWithQuestionMark`**

  The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `WildcardMinCharsWithAsterisk` property.

The properties that specify the minimum number of characters for Visual Query are:

- **`VisualQueryWildcardMinCharsWithAsterisk`**

  The minimum number of characters other than asterisks (*) that must be included in a Visual Query condition that contains or implies asterisks.

- **`VisualQueryWildcardMinCharsWithQuestionMark`**

  The minimum number of characters other than question marks (?) and asterisks (*) that must be included in a wildcard query that contains a question mark. This value should be less than, or equal to the value of the `VisualQueryWildcardMinCharsWithAsterisk` property.

If you follow this procedure for a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. For Quick Search, edit the values of the `WildcardMinCharsWithAsterisk` and `WildcardMinCharsWithQuestionMark` properties.
3. For Visual Query, edit the values of the `VisualQueryWildcardMinCharsWithAsterisk` and `VisualQueryWildcardMinCharsWithQuestionMark` properties.
4. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```

3. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```

4. Start Liberty:

   ```
   setup -t startLiberty
   ```

What to do next

Run a selection of Quick Search and Visual Queries that use conditions including wildcard characters. Continue to change the configuration until you are satisfied with the wildcard behavior.

Configuring Visual Query

Analysts can use Visual Query to search for records in the Information Store based on the values of specific properties and on their relationship to other records. You can configure a number of aspects of Visual Query, including the options that are available to analysts and how regularly saved queries are run.

Note: You can also configure how analysts complete Visual Queries by changing the minimum number of characters they must include in search terms that contain wildcards. For more information, see Modifying the wildcard minimum character limits.

Controlling the Visual Query palette

By default, the palette of query items that analysts see when they create and edit query structures contains all the item types in the i2 Analyze schema. To make the palette contain only the item types that you want analysts to use, you can edit the Visual Query configuration file.

Before you begin

Changing the contents of the Visual Query palette means adding the `<PaletteItemTypes>` element to the configuration file. For example, the `visual-query-configuration.xml` file in the `toolkit\configuration\live` directory includes a commented-out demonstration of using the element with the law enforcement schema:

```
<PaletteItemTypes Mode="DENY" ItemTypeIds="ET3, LAS1"/>
```

The `<PaletteItemTypes>` element has two mandatory attributes: `Mode` and `ItemTypeIds`. The value of the former determines how the server interprets the value of the latter:

- To specify exactly which item types the palette contains, use `Mode="ALLOW"`. Then, only the types whose identifiers are in the `ItemTypeIds` list appear in the palette that users see.
- To exclude certain item types specifically (but allow the palette to contain all other item types), use `Mode="DENY"`. Users do not see the types whose identifiers are in the `ItemTypeIds` list in the palette.

Important:

- It is not valid to specify operator usage rules for a denied type. Neither the identifier of a denied item type nor those of its property types must appear anywhere in the `<OperatorUsageRules>` element that the Visual Query configuration file also contains.
- If you exclude item types from the palette after analysts have started to use the system, any saved queries that use those item types become invalid. If analysts have configured alerts for those queries, they will no longer receive them. Always check with your analysts before you exclude item types from the palette.

About this task

Because it involves editing the Visual Query configuration file, the procedure for controlling the Visual Query palette and updating the server with your changes is similar to the one for restricting Visual Query conditions. See that topic for more information.

Procedure

To change the item types that users see in the Visual Query palette:

1. Open the `toolkit\configuration\live\visual-query-configuration.xml` file in an XML editor.
2. Using the supplied example as a guide, remove the comment from the `<PaletteItemTypes>` element and edit it according to your requirements.
3. Save your changes.
4. If you have not done so already, follow the instructions in Restricting Visual Query conditions to update `DiscoServerSettingsCommon.properties` with details of your configuration file.

   Follow the instructions in the final part of the same procedure to update the i2 Analyze deployment.

What to do next

Connect to i2 Analyze through either the i2 Notebook web client or the Analyst's Notebook desktop client. Verify that the item types available in the Visual Query palette are what you expect them to be.

Visual Query condition rules

Visual Query condition rules enable you to prevent users from running queries with particular characteristics, or to allow only specific queries. To manage the performance of Visual Query effectively, it is important to understand the types of rules that are available.

The sample `visual-query-configuration.xml` file in the `toolkit\configuration\live` directory includes commented-out examples of rules that can apply to the law enforcement schema. Using these examples as guidance, you can create rules that apply to your system.

Important:

- Any rules that you add to `visual-query-configuration.xml` (and subsequently deploy) affect the i2 Analyze server but *not* the Analyst's Notebook desktop client. As such, users are still able to construct queries that include restricted conditions, but the server will prevent such queries from being run.
- If you create rules that restrict specific characteristics after analysts have started to use the system, any saved queries that use those characteristics become invalid. If analysts have configured alerts for those queries, they will no longer receive them. Always check with your analysts before you restrict characteristics from Visual Queries.

Depending on your circumstances, there are two main approaches that can be used to create your rules:

- **Invalidating a particular type of query**

  If you know that your data can cause problems for particular queries - for example, a query that involves a particular item type or a particular operator - then you can prevent such queries from taking place. All other Visual Query behavior can then remain unchanged.
- **Enabling specific queries**

  If you are uncertain about which kinds of query might encounter problems, or you want to restrict the types of searches that users can carry out, you can start by preventing *all* queries, and then selectively allow the types of query that you require.

As you build up your Visual Query configuration file, bear in mind the following key concepts:

- **Rule components**

Each rule consists of the following components:

- Rule type - Whether the rule should allow or deny a specific type of query
- Item type - Which item type the rule should apply to
- Property type - Which property types of the specified item type the rule should apply to
- Operator - Which operators the rule should apply to
- Date and time aspects - Which types of temporal aspect the rule should apply to. For example, day of the week, or time of the day.

- **Implicit 'all'**

  If a rule does not specify one or more of the above components, then it applies to the component as a whole. For example, a rule that specifies an item type but not a property type applies to *all* property types of that item type.

- **Rule ordering**

  Visual Query rules are applied sequentially, which means that rules that come later in the file can override earlier. This allows you to refine your conditions. For example:

```
<Deny/>
<!-- Allow 'Person' searches to include conditions for 'Date of birth' and
 'Gender' that are exactly or between a specified range -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15" Operators="EQUAL_TO,
 BETWEEN"/>
```

Using these rules, searches for people with specified dates of birth or genders are permitted, but all other Visual Query conditions are prevented from running.

Restricting Visual Query conditions

By default, all the Visual Query operators are available for all the record types that support them in your system. To prevent the creation of queries that take too long to produce results, you can configure Visual Query to restrict the operators that are valid in conditions.

Before you begin

The Visual Query configuration is one of a group of configuration files that you can modify and send to the i2 Analyze server without the need for system downtime. These files are stored in the `toolkit\configuration\live` directory.

To enable updating the server without the need to restart it, ensure that you have a command-line tool such as `postman` or `curl`.

About this task

Visual Queries are closely aligned with the Information Store schema. You create and edit the Visual Query configuration file in your configuration development environment, alongside highlight queries and search results filters.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

Set the rules that you want to apply to Visual Query conditions:

1. Open the `toolkit\configuration\live\visual-query-configuration.xml` file in an XML editor.
2. Using the supplied examples as a guide, add the rules that you want to implement.

   Note: Pay attention to the order of your rules. Where there are clashes, later rules override earlier ones.
3. Save your changes.

Update the `DiscoServerSettingsCommon.properties` file to use your configuration rules:

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Set the `VisualQueryConfigurationResource` property to the name of the configuration file. For example:

   ```
   VisualQueryConfigurationResource=visual-query-configuration.xml
   ```
3. Save your changes.

Update the deployment with your changes. The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

Note: To redeploy your changes by using the deployment toolkit only, see Redeploying Liberty. You must use the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

1. At the command line, navigate to the `toolkit\scripts` directory.
2. Update the server with your configuration file:

   ```
   setup -t updateLiveConfiguration
   ```
3. Update the running application by using the reload endpoint:

   ```
   curl -i --cookie-jar cookie.txt -d j_username=<user_name>
                                   -d j_password=<password>
                                   http://<host_name>/<context_root>/
   j_security_check

   curl -i --cookie cookie.txt -X
       POST http://<host_name>/<context_root>/api/v1/admin/config/reload
   ```

   Warning: The `reload` method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

   The server validates the visual query configuration as it loads it, and returns any errors in its response.
4. If the configuration is invalid, modify the `visual-query-configuration.xml` file, and then repeat the process to update the deployment.

What to do next

Run a selection of Visual Queries that use conditions. Continue to change the configuration until you are satisfied with its behavior.

Rule type

The type of a Visual Query condition rule determines whether it specifically allows or denies conditions of a particular kind.

The type of a rule can be either:

- Deny, in which case the rule prevents conditions from being used
- Allow, in which case the rule permits conditions to be used

You *must* specify the type of each rule that you add to the Visual Query configuration file.

Item type identifier

A rule that includes only the rule type applies to all item types in the schema. To apply a rule to a specific item type, you can add an item type identifier.

You can use the `ItemTypeId` attribute to specify an item type whose conditions are subject to the rule. The identifier that you use must be present in the i2 Analyze schema.

To help with later troubleshooting, you might also add the display name or a description of the item type into a comment about the rule.

For example, given the following item type in the example law enforcement schema:

```
<EntityType Id="ET5"
            SemanticTypeId="guid8A586959-9837-47DE-8DBF-BC7031F01545"
            Description="Person details"
            DisplayName="Person"
            Icon="Person (Shaded Shirt)">
```

You might create the following rule in your configuration file:

```
<!-- Allow 'Person' searches to include conditions that specify exact values
 or ranges -->
<Allow ItemTypeId="ET5" Operators="EQUAL_TO, BETWEEN"/>
```

Note: You can specify only one item type per rule.

Property type identifiers

By default, when you add an item type identifier to a Visual Query condition rule, the rule applies equally to conditions involving all property types of that item type. To apply the rule only to specific property types, you can add property type identifiers.

The `PropertyTypeIds` attribute allows you to specify which property types the rule affects. The identifiers that you use must be present in the i2 Analyze schema.

To help with later troubleshooting, you might also add the display name or a description of the property types into a comment about the rule.

Note: Property type identifiers can only be used in rules that specify an item type, and the property types must correspond to the specified item type.

For example, given the following property types in the example law enforcement schema:

```
<EntityType Id="ET5"
            SemanticTypeId="guid8A586959-9837-47DE-8DBF-BC7031F01545"
            Description="Person details"
            DisplayName="Person"
            Icon="Person (Shaded Shirt)">
  ...
```

```
  <PropertyType Position="2"
                Mandatory="false"
                SemanticTypeId="guidFE45F1C4-B198-4111-8123-F42D2CD6419D"
                DisplayName="Date of Birth"
                Description=""
                LogicalType="DATE"
                Id="PER9">
    <PossibleValues/>
  </PropertyType>
  <PropertyType Position="3"
                Mandatory="false"
                SemanticTypeId="guid7548369B-BA9A-4C4B-AEAD-0CB442EAFA27"
                DisplayName="Gender"
                Description=""
                LogicalType="SUGGESTED_FROM"
                Id="PER15">
    <PossibleValues>
      <PossibleValue Description="" Value="&lt;Unknown&gt;"/>
      <PossibleValue Description="Male" Value="Male"/>
      <PossibleValue Description="Female" Value="Female"/>
    </PossibleValues>
  </PropertyType>
  ...
</EntityType>
```

You might create the following rule in your configuration file:

```
<!-- Allow 'Person' searches to include conditions involving 'Date of birth'
 and 'Gender' that specify exact values or range -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15" Operators="EQUAL_TO,
 BETWEEN"/>
```

Note: Because 'Gender' is a "suggested from" property type, there is no range of values for the BETWEEN operator to work with. Therefore, conditions that specify a range of values are enabled only for the 'Date of birth' property type.

Date and time aspects

When a rule specifies property types with the DATE or DATE_TIME data type, there are a number of options for querying that data. You can create rules that affect conditions that use specific aspects of temporal information.

DATE_AND_TIME

Apply the rule to conditions that include both a date and time. For example, you might specifically allow searches for people seen on a certain date at a certain time.

DATE

Apply the rule to conditions that focus on a date. For example, you might allow searches for people born on a particular day.

TIME

Apply the rule to conditions that focus on a time. For example, you might allow searches for financial transactions that regularly occur at a set time.

DAY_OF_MONTH

Apply the rule to conditions that focus on the day of the month. For example, you might allow searches for people paid on a specific day of the month.

MONTH

Apply the rule to conditions that focus on the month. For example, you might allow searches for people born in a particular month.

QUARTER

Apply the rule to conditions that focus on the quarter of the year. For example, you might allow searches for financial results.

YEAR

Apply the rule to conditions that focus on a specific year. For example, you might allow searches for people born in a particular year.

DAY_OF_WEEK

Apply the rule to conditions that focus on a specific day of the week. For example, you might allow searches for events that always occur on a Tuesday.

WEEK_OF_YEAR

Apply the rule to conditions that focus on the week of the year, as calculated using the ISO week date system. For example, you might allow searches for weekly sales results.

Operators

You can use operators to specify the types of Visual Query condition to constrain with a rule. If you do not specify an operator, the rule applies to all valid operations.

STARTS_WITH

For example, "Starts with: Fred" matches Fred, Frederick, or Freddie.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!-- Allow 'Person' searches to include conditions for 'First (Given) Name'
 that start with a given value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4" Operators="STARTS_WITH"/>
```

ENDS_WITH

For example, "Ends with: Fred" matches Fred, Wilfred, or Alfred.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!-- Allow 'Person' searches to include conditions for 'First (Given) Name'
 that end with a given value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4" Operators="ENDS_WITH"/>
```

CONTAINS

For example, "Contains: Fred" matches any of the above terms, and also matches Alfredo.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!-- Allow 'Person' searches to include conditions for 'First (Given) Name'
 that contain a given value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4" Operators="CONTAINS"/>
```

WILDCARD_PATTERN

An exact match for a specified term that includes wildcard characters. For example, "Wildcard pattern: Fr?d" matches Fred, but not Alfredo.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!-- Allow 'Person' searches to include conditions for 'First (Given) Name'
 that match a wildcard pattern -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4" Operators="WILDCARD_PATTERN"/
>
```

NOT_WILDCARD_PATTERN

A match for anything except for a specified term that includes wildcard characters. For example, "Not wildcard pattern: Fr?d" matches anything apart from Fr?d.

Supported logical types:

- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!-- Allow 'Person' searches to include conditions for 'First (Given) Name'
 that do not match a wildcard pattern -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER4"
 Operators="NOT_WILDCARD_PATTERN"/>
```

EQUAL_TO

An exact match for the specified term.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!-- Allow 'Person' searches to include conditions for  'Date of birth' and
 'Gender' that match an exact value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15" Operators="EQUAL_TO"/>
```

NOT_EQUAL_TO

Exact matches for the specified term are excluded. For example, "Not equal to: Fred" matches anything apart from Fred.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

Note: The server places a limit on the length of the strings that this operator considers. By default, the limit is 256 characters.

For example:

```
<!-- Allow 'Person' searches to include conditions for 'Date of birth' that
 do not match an exact value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9" Operators="NOT_EQUAL_TO"/>
```

GREATER_THAN

Matches values that are greater than a set value.

Supported logical types:

- DECIMAL

- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!-- Allow 'Person' searches to include conditions for 'Date of birth' that
 are greater than a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9" Operators="GREATER_THAN"/>
```

GREATER_THAN_OR_EQUAL_TO

Matches values that are greater than or equal to a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!-- Allow 'Person' searches to include conditions for 'Date of birth' that
 are greater than or equal to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
 Operators="GREATER_THAN_OR_EQUAL_TO"/>
```

LESS_THAN

Matches values that are less than a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!-- Allow 'Person' searches to include conditions for 'Date of birth' that
 are less than a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9, PER15" Operators="LESS_THAN"/
>
```

LESS_THAN_OR_EQUAL_TO

Matches values that are less than or equal to a set value.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!-- Allow 'Person' searches to include conditions for 'Date of birth' that
 are less than or equal to a specified value -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9"
 Operators="LESS_THAN_OR_EQUAL_TO"/>
```

BETWEEN

Matches values that are within a set range.

Supported logical types:

- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME

For example:

```
<!-- Allow 'Person' searches to include conditions for 'Date of birth' that
 match a range of values -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9" Operators="BETWEEN"/>
```

IS_SET

Matches properties with any populated value.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!-- Allow 'Person' searches to include searches for 'Date of birth' values
 that have been entered -->
```

```
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9" Operators="IS_SET"/>
```

IS_NOT_SET

Matches properties without a value entered.

Supported logical types:

- BOOLEAN
- DECIMAL
- DATE
- DATE_AND_TIME
- DOUBLE
- INTEGER
- TIME
- SINGLE_LINE_STRING
- SUGGESTED_FROM
- SELECTED_FROM

For example:

```
<!-- Allow 'Person' searches to include searches for 'Date of birth' values

that have not been entered -->
<Allow ItemTypeId="ET5" PropertyTypeIds="PER9" Operators="IS_NOT_SET"/>
```

Restricting the length of Visual Query lists

When a Visual Query is constructed by a user, certain conditions allow the values to be specified as a list. The length of these lists can be restricted to reduce the size of messages from the client.

About this task

By default, the number of items that can be used in a list condition for a visual query condition is set to 10000. If you reduce this value, the number of values that can be applied to conditions of this type is reduced.

To follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

To modify the list length:

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Edit the value of `VisualQueryMaxValuesInList`.
3. Save and close the file.

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

```
setup -t stopLiberty
```

3. Update the i2 Analyze application:

```
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

What to do next

Run a selection of Visual Queries that use conditions including lists. Continue to change the configuration until you are satisfied with the lists in your Visual Queries that use conditions.

Controlling Visual Query timeout

Depending on the complexity of the query structure and the volume of data involved, Visual Queries can take a long time to run. By default, i2 Analyze automatically terminates any Visual Query that runs for more than four hours, but you are free to increase or reduce this setting.

About this task

In the i2 Analyze configuration settings, the `VisualQueryTimeoutMinutes` property controls how long a Visual Query can run before the server stops it and reports that it failed.

If you know that your Visual Queries usually take less than four hours to run, then reducing the timeout can let you know sooner that something might be wrong. Alternatively, if you know that you have long-running queries that take more than four hours, you can increase the timeout.

To follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

To change the Visual Query timeout setting:

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Edit the value of `VisualQueryTimeoutMinutes`.
3. Save and close the file.

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

```
setup -t stopLiberty
```

3. Update the i2 Analyze application:

```
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

What to do next

Run a selection of Visual Queries and verify that under normal circumstances, none of them causes a timeout error. If necessary, increase the timeout setting until no predictable timeout errors occur.

Visual Query schedule

When a user enables alerting on a saved Visual Query, the query is run automatically on a schedule. The default schedule can be changed to suit your performance requirements and the number of Visual Queries that are saved with alerting enabled.

By default, the Visual Queries that are saved with alerting enabled are run once every 24 hours, at 00:00 according to the server time. If the result of a query is different from the last time that the query ran, the user receives an alert with this information.

You can change or override the default schedule, as follows.

- Change the time of day that the saved Visual Queries run. For more information, see Changing the daily schedule for Visual Queries.
- Create a custom schedule to specify when the saved Visual Queries run. For more information, see Creating a custom schedule for Visual Queries.

  Note: Running Visual Queries on schedule might affect performance for users of the system. The following factors are examples of the characteristics of Visual Queries that can affect performance:

  - Number of queries
  - Complexity of the queries
  - Volume of data searched by the queries

  In this situation, consider setting the daily schedule for a quiet period such as overnight, or setting the custom schedule for a low frequency.

Changing the daily schedule for Visual Queries

You can change the time of day that i2 Analyze runs Visual Queries that are saved with alerting enabled. By default, these queries run once every 24 hours at 00:00 according to the server time.

About this task

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Edit the value of `AlertScheduleTimeOfDay`.

   The default value is 00:00. The required format is *`HH:mm`*. *`HH`* is the hour of the day in a 24-hour time format, 00 to 23; and *`mm`* is the number of minutes past the hour, 00 to 59.
3. To ensure that the daily schedule value is implemented, check that the `AlertScheduleExpression` property is not enabled. For more information, see Creating a custom schedule for Visual Queries.
4. Save and close the file.

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```
4. Start Liberty:

   ```
   setup -t startLiberty
   ```

After you redeploy i2 Analyze, the Visual Queries that are saved with alerting enabled are run according to the updated daily schedule.

Creating a custom schedule for Visual Queries

To create a custom schedule to run the Visual Queries that are saved with alerting enabled, use a UNIX cron expression. The default schedule to run these Visual Queries is once every 24 hours, at 00:00 hours according to the server time.

About this task

You can create a custom schedule in the `AlertScheduleExpression` property. The `AlertScheduleExpression` property requires a value in UNIX cron expression format.

With the `AlertScheduleExpression` property, you can override the default daily schedule. By default, this property is not enabled. If the property *is* enabled, whenever the value of the expression matches the current date and time, i2 Analyze runs the Visual Queries that are saved with alerting enabled. For more information, see UNIX cron format.

When you enable the `AlertScheduleExpression` property, its value overrides the value of the default property, `AlertScheduleTimeOfDay`. For more information about `AlertScheduleTimeOfDay`, see Changing the daily schedule for Visual Queries.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Using a text editor, open the `DiscoServerSettingsCommon.properties` file. You can find this file in the following location: `toolkit\configuration\fragments\opal-services\WEB-INF\classes`.
2. Uncomment `AlertScheduleExpression`.

   When you uncomment the `AlertScheduleExpression` property, the value for this property is used instead of the value for the `AlertScheduleTimeOfDay` property.
3. Edit the value of `AlertScheduleExpression`.

   For example, the value `0,4,16,20 * * *` schedules the Visual Queries that are saved with alerting enabled to run every 4 hours. A short way of expressing this schedule is `0 */4 * * *`.
4. Save and close the file.

Redeploy i2 Analyze to update the application with your changes:

1. In a command prompt, navigate to the `toolkit\scripts` directory.

2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```

3. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```

4. Start Liberty:

   ```
   setup -t startLiberty
   ```

After you deploy i2 Analyze, the updated schedule is applied to run Visual Queries that are saved with alerting enabled.

Highlight queries

In i2 Analyze, *highlight queries* can automatically navigate the data in the Information Store on the behalf of users, and present them with answers to the questions that they ask most frequently.

Highlight queries run when a user of i2 Analyze client software chooses a subject record. For example:

- If the subject record is a car, then a highlight query might find its recent owners or its recent sightings.
- If the subject is a cellphone, then a highlight query might find which other phones it calls most often - or *who owns* the phones that it calls most often.
- If the subject is a person, then a highlight query might return their bank accounts; or the bank accounts that the person's accounts transact with; or the owners of those transacting accounts.

Technical background

Functionally, a highlight query is a predefined, type-specific, multi-level, conditional expand operation. Using the subject entity record as a seed, a highlight query searches the Information Store for entity records of a particular type that are connected to the seed by links of a particular type. After that, it can use those records as the seeds for another, similar search, and so on. The results that users see are the entity records found by the final search that the highlight query performs - which might be the first, or the second, or the sixth.

Because highlight queries are type-specific, they are tightly bound to the Information Store schema. Each entity type in the schema can have its own group of highlight queries that run when the user chooses a record of that type. Since highlight queries are targeted at finding specific relationships in your data, they are not typically portable between different entity types or different i2 Analyze deployments.

User experience

You can categorize highlight queries so that different users see different ones, but as a feature highlight queries are available to all i2 Analyze users. However, the user experience depends a little on which client software they're using.

- In the i2 Enterprise Insight Analysis Investigate Add On, users can select records from search results or from lists of records that they flagged.

  Selecting a record opens a view that contains the information in the subject, plus a set of *highlight panes* that display the first five results from each highlight query for that record's type. Each highlight pane also contains a button that the user can click to show more information and more results.

  Alongside the highlight panes is a visualization of the results they contain, and users can click records in the visualization to navigate the results.

- In the i2 Notebook web client and the Analyst's Notebook desktop client, the same functionality is available through the *360 view* feature, which complements the chart and the Record Inspector.

  To choose a record as the subject of a 360 view, users can select it from search results or use the right-click pop-up menu from a chart item. The 360 view opens in a separate panel or window from the rest of the application.

In all cases, the highlight queries that you define are key to the user experience. It's likely that you'll work with users to create the highlight queries that they need.

Automatically generated highlight queries

When a deployment of i2 Analyze starts up for the first time, the server automatically generates a set of highlight queries from the Information Store schema. This set contains one highlight query for each entity type in the schema that appears in at least one list of valid link end types. Each highlight query searches for entity-link-entity structures in which the types of the target link and entity are the first valid types for such structures.

For example, imagine a simple schema that defines four entity types (Address, Car, Cellphone, and Person) and four link types (Calls, Owns, Related, and Resides). According to the link type definitions:

- Cars can be linked to addresses, through Resides links
- Cellphones can be linked to other cellphones, through Calls links
- People can be linked to addresses (through Resides links), to cars and cellphones (through Owns links), and to other people (through Related links)

In this scenario, with the automatically generated highlight queries in place, a user who selects a cellphone record sees a highlight pane containing a list of cellphones that the first cellphone has called. (Calls is the only valid link type for cellphones, and Cellphone is the only valid end type.)

If the user selects a person record, they see a highlight pane containing a list of the cars that the person owns. (Owns is the *first* valid link type for people, and Car is the *first* valid end type for such links.)

Custom highlight queries

The automatic mechanism for generating highlight queries is designed only to be a starting point. When you write your own highlight queries, you can make them search for more complex relationships between the subject record and others in the Information Store.

For example, for data that matches the same schema as before, you might enable users to select a person record and see a list of people who reside at the same address (person-address-*person*), or to whom the subject makes cellphone calls (person-cellphone-cellphone-*person*).

To create a set of custom highlight queries for your data and your users, you write an XML configuration file that is valid according to an XSD file that is derived from the Information Store schema. Subsequent topics describe how to start, edit, and publish a highlight queries configuration for your deployment of i2 Analyze.

Deploying highlight queries

New, production deployments of i2 Analyze have a set of automatically generated highlight queries that retrieve records with simple relationships to a subject record. To replace this set with custom queries

that reflect the structure of your data and the needs of your users, you must write your own highlight queries configuration file.

Before you begin

The highlight queries configuration is one of a group of configuration files that you can modify and send to the i2 Analyze server without the need for system downtime. These files are stored in the `toolkit\configuration\live` directory. The default `highlight-queries-configuration.xml` file instructs the server to generate the automatic highlight query set.

To enable updating the server without the need to restart it, ensure that you have a command-line tool such as `postman` or `curl`.

About this task

The procedure for writing a new highlight queries configuration file starts with a download of two files from the server. Through a pair of REST endpoints, you can retrieve an XML file that defines the same set of highlight queries that the server generated automatically, and an XSD file that validates it. With these files as a guide, you can develop your own highlight queries, replace the default file, and upload it to the i2 Analyze server.

Highlight queries are closely aligned with the Information Store schema. You create and edit the highlight queries configuration file in your configuration development environment, alongside Visual Queries and search results filters.

Note: As described in Example highlight queries, example deployments of i2 Analyze receive an example set of highlight queries instead of the automatically generated set. To replace the example set with an automatically generated set, replace the highlight queries configuration file in `toolkit\configuration\live` with its equivalent from a `configuration\live` directory under the `toolkit\examples` directory. Then, pick up the procedure below at Update the deployment with your changes.

Procedure

The first part of the process is to fetch the XSD file that corresponds to the Information Store schema from the server.

1. Open a web browser and connect to the i2 Analyze server at `http://<host_name>/opal/doc` to view the REST API documentation.

   If you are not logged in to the server, you are prompted to do so. The user must be a member of a group that has the `i2:Administrator` permission under command access control.

   Note: This requirement is in addition to the other requirements on i2 Analyze users, all of whom must be members of groups that confer an access level for at least one dimension value in each security dimension.

2. Navigate to the `GET /api/v1/admin/highlightqueryconfig/xsd` method, and click **TRY** to request the XSD file from the server.

   The file is displayed in the **RESPONSE** field.

3. Save the contents of the field to a file named `highlight-queries-configuration.xsd` in the `toolkit\configuration\live` directory.

Next, you need a highlight queries configuration file to edit. To start from scratch, you can use the `highlight-queries-configuration.xml` file from the `live` directory. Alternatively, you can download the file that defines the automatically generated highlight queries.

1. Navigate to the `GET /api/v1/admin/highlightqueryconfig/automatic` method, and click **TRY** to request the XML file from the server.

   The file is displayed in the **RESPONSE** field.
2. Save the contents of the field to a file named `highlight-queries-configuration.xml` in the `toolkit\configuration\live` directory, replacing the existing file.

Now, edit the configuration file.

1. If you do not already have one, obtain and configure an XSD-aware XML editor, as described in Setting up your XSD-aware XML editor.
2. In the XML editor, open the `toolkit\configuration\live\highlight-queries-configuration.xml` file.
3. Using the reference and example information, modify the file to define the highlight queries that your users require.

Update the deployment with your changes. The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

Note: To redeploy your changes by using the deployment toolkit only, see Redeploying Liberty. You must use the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

1. At the command line, navigate to the `toolkit\scripts` directory.
2. Update the server with your configuration file:

   ```
   setup -t updateLiveConfiguration
   ```
3. Update the running application by using the reload endpoint:

   ```
   curl -i --cookie-jar cookie.txt -d j_username=<user_name>
                                   -d j_password=<password>
                                   http://<host_name>/<context_root>/
   j_security_check

   curl -i --cookie cookie.txt -X
       POST http://<host_name>/<context_root>/api/v1/admin/config/reload
   ```

   Warning: The `reload` method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

   The server validates your highlight queries as it loads them and returns any errors in its response.
4. If the configuration is invalid, modify the `highlight-queries-configuration.xml` file, and then repeat the process to update the deployment.

Test the new and updated highlight queries.

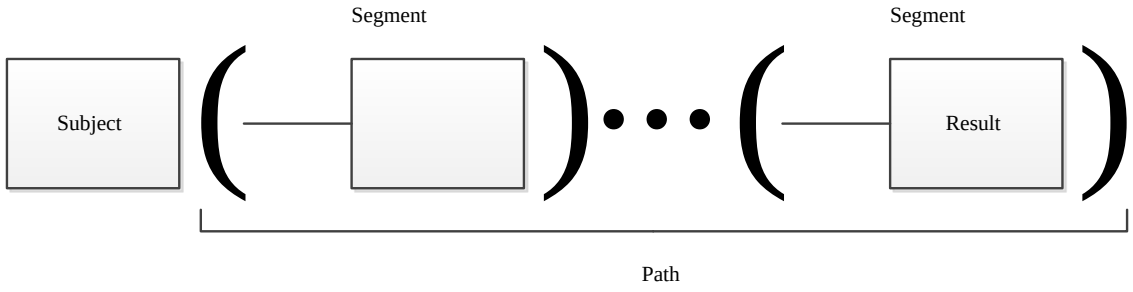
1. In your web browser, view the highlight panes for a record with a number of connections that test your highlight queries.

Structure of highlight queries

The aim of highlight queries is to provide users with relevant information about the records that are associated with the current subject of their investigation. The structure of the queries that you create reflects the idea of starting from the subject and finding records that are one, two, or more links away.

When a user examines records of the same type, they expect to be able to compare them in a consistent way. Highlight queries are grouped by type in the configuration file, and records of a particular type are always displayed with the same set of highlight panes for the same user.

Highlight queries themselves use the idea of a *path* to describe how the results of the query are related to the subject. The path is made up of *segments* that take the results one step further from the subject, like this:



At a minimum, each segment in a highlight query specifies the link type and the entity type of the records that match it. For example, a highlight query for finding people who live at the same address as the subject person might have two segments:

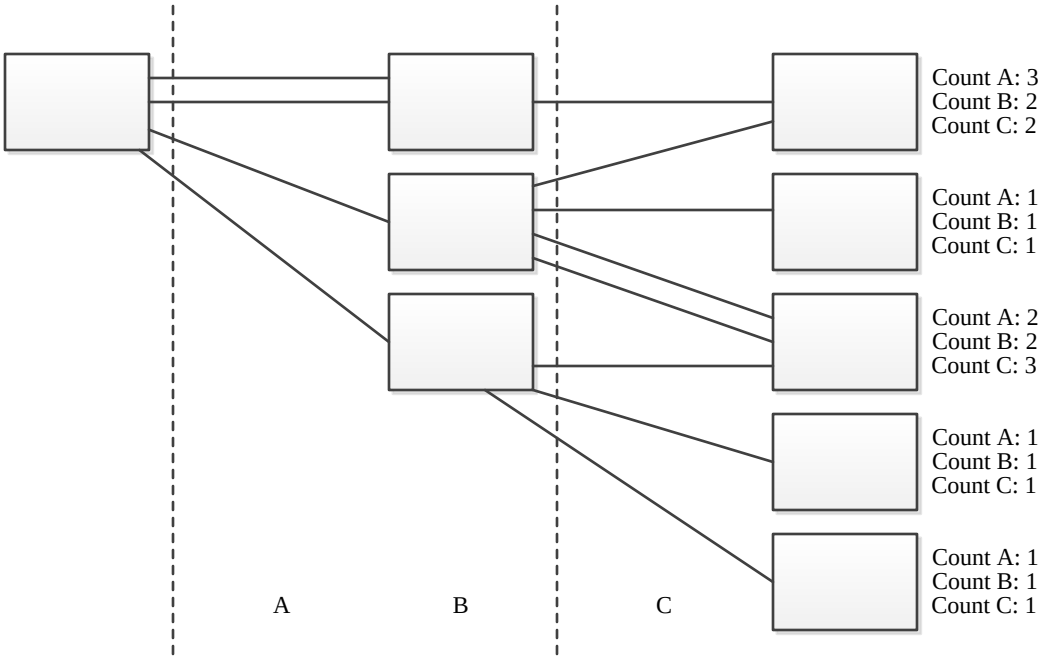
- A link of type 'Lives At' and an entity of type 'Address'. If this segment was the last one in the path, the results would be of type Address, and users might see a list of past and current addresses for the subject.
- A link of type 'Lives At' and an entity of type 'Person'. The records found in the first segment are used as inputs to the second segment, so the full query returns a list of all the people who are known to have lived at the same addresses as the subject.

You can further constrain the results of a highlight query by placing conditions on the link and entity records in the Information Store that each segment matches. Depending on the schema, you might decide that you are only interested in addresses that the subject has 'Lived At' in the last three years, or when the 'Address' is in a particular city or country. Similarly, you might only want to find people who are female (or, alternatively, male).

As well as controlling which records you want to find with a highlight query, you can control how users see the query results in highlight panes. Every query has a name and a description that users can read to understand what the results in front of them mean. For the results themselves, you can specify how to sort them, and what values to display in the limited space that highlight panes provide.

The values that users see in highlight panes do not have to come from the records that form the results of the query. In each segment of a highlight query, you can *export* values from the link and entity records that form part of a found path. For example, imagine a highlight query for bank account records whose single segment finds other accounts that are connected through transaction links. You can export the value of the largest transaction from the segment, and then set it as one of the outputs from the path.

Finally, as well as exporting property values from the segments of a highlight query, you can export (and subsequently output) the counts of the records that the highlight query finds at each point on the path.



In the diagram, the five records on the right are results of a highlight query whose subject was the single record on the left. When you present the results to users, you can output or sort by the counts of records along their paths. The meanings of the counts, and whether they are useful, vary according to the specifics of the highlight query.

The highlight queries configuration file

The permitted XML elements in a highlight queries configuration file include some that are common to all such files, and others that depend upon the Information Store schema. The structure of the file reflects the structure of the highlight queries themselves.

Root element

- `<highlightQueryConfiguration>`

Top-level elements

- `<settings>`
- `<categoryDefinitions>`
- `<highlightQueryGroups>`

Category definition elements

- `<categoryDefinition>`
- `<allow>`
- `<userGroup>`

Highlight query elements

- `<highlightQueryGroup>`
- `<highlightQuery>`
- `<categories>`
- `<description>`
- `<path>`
- `<sortBy>`

Path elements

- `<segments>`
- `<segment>`
- *`<EntityType>`*/*`<LinkType>`*
- `<direction>`
- `<conditions>`
- `<exportFields>`
- `<propertyField>`
- `<countField>`
- `<outputs>`

Condition elements

- *`<PropertyType>`*

Root element

`<highlightQueryConfiguration>`

`<highlightQueryConfiguration>` is the root element of the highlight query configuration file. In the examples, and in the automatically generated file, it references the `highlight-queries-configuration.xsd` file:

```
<highlightQueryConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:noNamespaceSchemaLocation="highlight-queries-configuration.xsd">
  ...
</highlightQueryConfiguration>
```

Top-level elements

`<settings>`

As well as the highlight queries themselves, the configuration file enables control of some aspects of highlight query behavior that can affect the performance of the i2 Analyze server. The optional `<settings>` element supports the following child elements:

- `<maxLinkResultsPerSegment>` - The maximum number of link records that a single segment can match in the Information Store, which defaults to 50000.
- `<maxSeedsPerSegment>` - The maximum number of entity records that any segment except the last can match in the Information Store, which defaults to 1000.
- `<maxResultsPerQuery>` - The maximum number of entity records that the last segment can match in the Information Store, which defaults to 1000.

- `<expireResultsInMinutes>` - The length of time for which the server caches the results of a highlight query, which defaults to 30 minutes.

```
<highlightQueryConfiguration ...>
  <settings>
    <maxLinkResultsPerSegment>50000</maxLinkResultsPerSegment>
    ...
  </settings>
  ...
</highlightQueryConfiguration>
```

If any of the "`max`" limits is breached, the query in question returns no results and the user sees an error message. Other queries that do not breach the limits continue to run normally.

`<categoryDefinitions>`

Optionally, you can control which users see a highlight query by putting it into one or more *categories*. A category specifies the user groups whose members can see the highlight queries it contains. To use categories, you define them at the top of the highlight queries configuration file, and then apply them to highlight query definitions.

When present, the `<categoryDefinitions>` element is a child of the `<highlightQueryConfiguration>` element:

```
<highlightQueryConfiguration ...>
  <settings>...</settings>
  <categoryDefinitions>
    ...
  </categoryDefinitions>
  ...
</highlightQueryConfiguration>
```

When the `<categoryDefinitions>` element is not present, all highlight queries are visible to all users who can see the types for which they're defined.

`<highlightQueryGroups>`

The only other element permitted as a direct child of `<highlightQueryConfiguration>` is `<highlightQueryGroups>`, which contains all the highlight queries for the deployment, grouped by the entity types of the subject records to which they apply.

```
<highlightQueryConfiguration ...>
  <settings>...</settings>
  <categoryDefinitions>...</categoryDefinitions>
  <highlightQueryGroups>
     ...
  </highlightQueryGroups>
</highlightQueryConfiguration>
```

Category definition elements

`<categoryDefinition>`

The `<categoryDefinitions>` element can contain any number of `<categoryDefinition>` children, each of which must have a name and can have a list of "allowed" user groups.

```
  <categoryDefinitions>
    <categoryDefinition name="Finance">
      <allow>
        <userGroup name="Analyst"/>
        <userGroup name="Clerk"/>
      </allow>
```

```
    </categoryDefinition>
    ...
  </categoryDefinitions>
```

The `name` of each category definition must be unique; it's used later in the file to put highlight queries into categories. A category definition without a child `<allow>` element can still be used to categorize, but has no effect on the visibility of highlight queries.

`<allow>`

The single, optional `<allow>` element in each category definition specifies an OR'ed list of the user groups whose members can see the queries in the category. An empty `<allow>` element is equivalent to an absent one.

`<userGroup>`

The `<userGroup>` elements that form the list of allowed user groups have a single `name` attribute that must match the name of an i2 Analyze user group. This dependency is validated at server startup and when the live highlight queries configuration file is modified.

Highlight query elements

`<highlightQueryGroup>`

Each highlight query group contains the highlight queries for subject records of a particular type. The `itemType` attribute of the `<highlightQueryGroup>` element indicates which type the group is for. The valid values for the `itemType` attribute are defined in the XSD file; they are based on the display names of entity types in the Information Store schema.

```
  <highlightQueryGroups>
    <highlightQueryGroup itemType="Vehicle-I">
      ...
    </highlightQueryGroup>
    ...
  </highlightQueryGroups>
```

`<highlightQuery>`

A `<highlightQueryGroup>` element can have multiple child `<highlightQuery>` elements. Each `<highlightQuery>` element defines a query to run against the Information Store, and controls some aspects of the highlight pane that displays the query results in the user interface.

The `<highlightQuery>` element supports the `title` attribute whose value is displayed as the title of the highlight pane. It also supports the `automatic` attribute that determines whether the query runs as soon as a user views the subject record. You can specify that users must run a time-consuming query manually (by clicking a button) by setting the value of this optional attribute to `false`.

`<categories>`

The optional `<categories>` child of a `<highlightQuery>` element is the mechanism through which you put highlight queries into categories. If the `<categories>` element is absent or empty, then the query is not in a category, and it is available to all users.

All `<category>` children of the `<categories>` element must have a `name` attribute whose value matches the name of an earlier `<categoryDefinition>`.

When the highlight queries configuration file is processed, the categories are OR'ed together. The resulting behavior is that a highlight query is visible to users who are in *any* of the groups that appear in the definitions of *any* of the categories in this list.

`<description>`

In the <description> child of a <highlightQuery> element, you can explain the behavior of the query to users. The contents of this element are displayed to users when they click the information icon **(i)** in a highlight pane heading. At a minimum, i2 recommends that you describe the relationship of the results to the subject record, and the order that you chose to present them in.

<path>

The <path> child of a <highlightQuery> element contains <segments> and <outputs> elements that together provide the logic for the query. Segments are responsible for the query structure, while outputs determine how users see the query results in highlight panes.

<sortBy>

The last permitted child of a <highlightQuery> element is <sortBy>. This optional (but important) element specifies how values that were discovered during execution of the query can change the results that users see. For example, depending on your investigation, you might sort a list of people who are associated with a vehicle by their family names or by their dates of association.

The <field> children of <sortBy> are processed in order of appearance. In each one, the id attribute contains the identifier of a field that was exported from the path. The order attribute determines whether results appear in ascending (ASC) or descending (DESC) order of the values of the identified field.

```
<highlightQueryGroup itemType="Vehicle-I">
  <highlightQuery title="People with access" automatic="true">
    <categories>
      <category name="Finance"/>
      ...
    </categories>
    <description>People who have access to this vehicle,
      sorted by family name.</description>
    <path>...</path>
    <sortBy>
      <field id="familyName" order="ASC"/>
    </sortBy>
  </highlightQuery>
  ...
</highlightQueryGroup>
```

Path elements

<segments>

<path> elements always have exactly two children: <segments> and <outputs>. <segments> always contains at least one element that defines a part of the query structure.

```
<path>
  <segments>
    <segment>
      ...
    </segment>
    ...
  </segments>
  <outputs>
    ...
  </outputs>
</path>
```

****

Each element also has exactly two children that specify the link and entity types that records must have in order to match this part of the query. Like the legal values for the itemType attribute of the <highlightQueryGroup> element, the names of these elements are not fixed but depend upon the Information Store schema for the deployment. Furthermore, the link type must be valid for the entity type that you specified in <highlightQueryGroup>, and the entity type must be valid for that link type. For example, a segment for people who have access to vehicles:

```
<highlightQuery title="People with access">
  <path>
    <segments>
      <segment>
        <Access_To-I/>
        <Person-I/>
      </segment>
      ...
```

It is valid to use <AnyLinkType/> in place of a specific link type from the schema. It is also valid to use <AnyEntityType/> in place of a specific entity type, in any segment except the last one in the path.

<EntityType>/<LinkType>

The previous example contains a functionally complete segment definition. In a highlight query, the segment finds all the people in the Information Store that are connected to the inputs by 'Access To' links. If the segment is the first in the path, its single input is the subject record. Otherwise, its inputs are the records that the previous segment found.

The "entity type" and "link type" elements that make up a segment support three child elements, all of which are optional. To place further constraints on the records that the segment matches, you can add a <direction> or a <conditions> element. If you want to use some values from the records that you find at this stage of the query to provide context for the final results, you can add an <exportFields> element.

<direction>

For link type elements only, you can constrain matching records according to whether their direction is towards or away from the entity record that forms the input to the segment. The <direction> child of a link type element always contains a single <value> child element whose text content is either INCOMING or OUTGOING.

A constraint of this type is often useful in scenarios such as telephone call analysis or financial investigations, where the direction of a link is key to determining its relevance.

<conditions>

The <conditions> child of an entity type or link type element can contain any number of child elements that correspond to property types of the entity type or link type in question. For each property type, you can place a requirement on the values that matching records must have.

Note: At this version of i2 Analyze, highlight queries do not support conditions on property types with geospatial or multi-line string logical types.

<exportFields>

You might want to export values from a segment of a highlight query for two reasons. First, to display those values to users alongside the results in the highlight pane for the query. Second, to use the values as criteria by which to sort the results of the query.

For a particular highlight query result, the values that you can export are the values of properties from the segments that contributed to that result. For example, imagine a query that finds bank account records that are connected through transaction links. You want to display the transaction dates from

those links alongside the found accounts. In this example, every account is connected to the subject record through different links from every other account. If you export the date from the link type, each result sees a date value that comes from its particular links.

```
<segments>
  <segment>
    <Access_To-I>
      <direction>...</direction>
      <conditions>...</conditions>
      <exportFields>
        ...
      </exportFields>
    </Access_To-I>
    <Person-I/>
  </segment>
  ...
```

The `<exportFields>` element supports any number of two child element types: `<propertyField>` and `<countField>`.

`<propertyField>`

The `<propertyField>` child of the `<exportFields>` element has three mandatory attributes:

- `propertyType` - The property type whose value you want to export from the segment. Valid values for this attribute depend on the current entity or link type and are defined in the XSD file. These values are based on the display names of property types in the Information Store schema.
- `aggregatingFunction` - For any particular result, it is possible for a segment to match multiple records. The aggregating function specifies what property value to use when that situation arises. Set the attribute value to `MAX` or `MIN` to select the highest or lowest property value from such a set. Alternatively, set the value to `SUM` to add the property values from all the matching records together.
- `id` - An identifier for the exported value that you can use to refer to it from elsewhere in the highlight query structure.

```
<Access_To-I>
  <direction>...</direction>
  <conditions>...</conditions>
  <exportFields>
    <propertyField id="typeOfUse" propertyType="type_of_use-P"
      aggregatingFunction="MAX"/>
    ...
  </exportFields>
</Access_To-I>
```

`<countField>`

As well as using `<propertyField>` to export the highest, lowest, or total value of a property from a set of records, you can use `<countField>` to export the number of records in that set. Returning to the example of linked accounts, it might be useful to tell users how many transactions took place between the subject and each result, or to sort the results by that number.

The `<countField>` element has a single mandatory attribute: `id`, which behaves exactly as it does in `<propertyField>`.

`<outputs>`

After the `<segments>` in the path are the `<outputs>`, which control the values that users see with the query results in a highlight pane. The `<outputs>` element supports any number of child `<field>` elements, which have two attributes:

- `id` - The identifier of an export field, as previously specified in a `<propertyField>` or `<countField>` element.
- `label` - A value that the user interface uses as a column header for the field.

```
<path>
  <segments>
    <segment>
      ...
    </segment>
    ...
  </segments>
  <outputs>
    <field id="typeOfUse" label="Type of use" />
    ...
  </outputs>
</path>
```

Each highlight pane has space for up to two output fields. If you specify more than two `<field>` elements, users see them only when they click **Show more** in the highlight pane to display more result information.

Condition elements

<PropertyType>

Inside an *<EntityType>* or *<LinkType>* element, the `<conditions>` element can contain any number of child elements whose names are based on property types from the Information Store schema. The child elements restrict the records in the Information Store that match the current segment. The permitted contents of the condition depend on the logical type of the property type.

```
<Access_To-I>
  <conditions>
    <type_of_use-P>
      ...
    </type_of_use-P>
    ...
  </conditions>
  ...
</Access_To-I>
```

Note: Conditions are valid in *every* segment of a highlight query. Adding a condition to the entity type element in the final segment has the effect of directly filtering the results that a user sees.

Some logical types - geospatial values, multiple-line strings, documents, pictures, and XML data - are not supported in highlight query conditions. According to the generated XSD file, property types with unsupported logical types are not valid children of `<conditions>` elements. For property types with supported logical types, this version of i2 Analyze supports conditions with four kinds of operators:

- For all valid property types, you can use `<isSet>` and `<isNotSet>` elements to specify that matching records must (or explicitly must not) have a value for the property with that type.

```
<conditions>
  <type_of_use-P>
    <isSet/>
  </type_of_use-P>
  ...
</conditions>
```

- For property types whose properties have string values, you can use `<equalTo>` and `<notEqualTo>` elements to specify that matching records must have a value for the property that exactly matches (or explicitly does not match) values that you provide.

```
<conditions>
  <type_of_use-P>
    <notEqualTo>
      <value>Passenger</value>
      ...
    </notEqualTo>
  </type_of_use-P>
  ...
</conditions>
```

- For property types whose properties have numeric values, you can use `<greaterThan>`, `<lessThan>`, `<greaterThanOrEqualTo>`, and `<lessThanOrEqualTo>` elements to specify that matching records must have a property value with the specified relationship to a value that you provide.

```
<conditions>
  ...
  <transaction_value-P>
    <greaterThan>
      <value>50000</value>
    </greaterThan>
  </transaction_value-P>
  ...
</conditions>
```

- For property types whose properties have date and time values, you can use an `<inThePreceding>` element to specify that matching records must have a value for the property that occurred within a certain period before the current date.

```
<conditions>
  <type_of_use-P>
    ...
  </type_of_use-P>
  <end_date_and_time-P>
    <inThePreceding unit="WEEKS">
      <value>6</value>
    </inThePreceding>
  </end_date_and_time-P>
  ...
</conditions>
```

  In `<inThePreceding>` elements, the `<value>` must be an integer, while the `unit` attribute can be one of `WEEKS`, `MONTHS`, or `YEARS`.

If you specify multiple property types in the same `<conditions>` block, their effects are ANDed together. For example, if your conditions are "given name equal to 'John'" and "family name equal to 'Doe'", then only records for which both conditions are true appear in the results.

Conversely, if you use multiple operators against the same property type, their effects are ORed together. For example, you might decide that a property must either be not set or have one of a handful of values. However, you cannot use the same operator more than once, and you cannot use more than one of the numeric operators at the same time. Attempting to do either results in an invalid configuration file.

Example highlight queries

An installation and deployment of i2 Analyze provides two sets of example highlight queries that you can use alongside the documentation when you write your own queries.

The first set of example highlight queries is in the i2 Analyze deployment toolkit. The `toolkit\examples\highlight-queries\highlight-queries-configuration.xml` file contains an annotated set of highlight queries that are compatible with the example law enforcement schema. In an example deployment of i2 Analyze, these queries take the place of the automatically generated set.

The second set of example highlight queries is in the XML that you can fetch from a running i2 Analyze deployment. These automatically generated queries are compatible with the live Information Store schema.

A worked example

The following listing represents one of the more complicated highlight queries in the toolkit example:

```
<highlightQuery title="Money laundering?" automatic="false">
  <description>Organizations whose accounts transact with an account that
  also transacts with this person's accounts (and might therefore be an
  intermediary), sorted by value of the person's transactions.</description>
  <path>
    <segments>
      <segment>
        <Access_To-I/>
        <Account-I/>
      </segment>
      <segment>
        <Transaction-I>
          <conditions>
            <transaction_currency-P>
              <equalTo>
                <value>US Dollars</value>
              </equalTo>
            </transaction_currency-P>
          </conditions>
          <exportFields>
            <propertyField propertyType="transaction_value-P"
                           aggregatingFunction="MAX" id="value"/>
            <propertyField propertyType="date_and_time-P"
                           aggregatingFunction="MAX" id="dateTime"/>
            <countField id="transactioncount"/>
          </exportFields>
        </Transaction-I>
        <Account-I/>
      </segment>
      <segment>
        <Transaction-I>
          <exportFields>
            <countField id="transactioncount2"/>
          </exportFields>
        </Transaction-I>
        <Account-I/>
      </segment>
      <segment>
        <AnyLinkType/>
        <Organization-I/>
      </segment>
    </segments>
```

```
    <outputs>
      <field label="# txns Per -- Inter" id="transactioncount"/>
      <field label="# txns Inter -- Org" id="transactioncount2"/>
      <field label="Largest txn value" id="value"/>
      <field label="Most recent txn date" id="dateTime"/>
    </outputs>
  </path>
  <sortBy>
    <field id="value" order="DESC"/>
  </sortBy>
</highlightQuery>
```

This highlight query is for subject records of type 'Person'. There is no `<categories>` element, so the query is available to all users who have permission to see 'Person' records.

The results that the query finds are of type 'Organization', because that is the entity type in the final segment. There are four segments in all, which has the potential to make the query resource-intensive. The enclosing `<highlightQuery>` element has its `automatic` attribute set to `false` so that the query only runs when a user requests it.

The first segment in the query finds all the 'Account' records in the Information Store to which the subject is connected through an 'Access To' link. There are no conditions on either part of the segment, and no values are exported for use elsewhere.

The second segment takes all the accounts to which the subject has access, and finds accounts that they have exchanged transactions with. However, it does not find all such accounts, because of the condition on the link type:

```
<conditions>
  <transaction_currency-P>
    <equalTo>
      <value>US Dollars</value>
    </equalTo>
  </transaction_currency-P>
</conditions>
```

The condition restricts the accounts that this segment finds to those where transactions have taken place in US dollars. ("`US Dollars`" is configured in the Information Store schema as a possible value for properties of this type.) The next part of the segment then exports information about these dollar transactions for later use:

```
<exportFields>
  <propertyField propertyType="transaction_value-P"
                 aggregatingFunction="MAX" id="value"/>
  <propertyField propertyType="date_and_time-P"
                 aggregatingFunction="MAX" id="dateTime"/>
  <countField id="transactioncount"/>
</exportFields>
```

For each of the eventual results of the highlight query, these lines record the value of the largest transaction at this location in the path, and the date of the most recent transaction. These values might come from different transactions if the count - which we also export here - is greater than one.

The inputs to the third segment, then, are all the accounts that have transacted in dollars with accounts to which the subject has access. The third segment goes on to find any accounts in the Information Store that have exchanged transactions with the inputs. It also exports the count of transactions at this location in the path:

```
<exportFields>
  <countField id="transactioncount2"/>
</exportFields>
```

The final segment takes these accounts that are twice-removed from accounts to which the subject has access, and finds organizations in the Information Store that are connected to them in any way. To do so, it makes use of an `<AnyLinkType>` element:

```
<segment>
  <AnyLinkType/>
  <Organization-I/>
</segment>
```

It is worth recounting what this means from an investigative point of view. For an organization to be found by this query, it must be linked to an account that has transacted with an account that has also exchanged dollar transactions with an account to which the subject has access. In simpler terms, it might be that accounts belonging to the person and the found organizations are exchanging money through third accounts. The query certainly is not conclusive, but the results might become targets for further investigation.

Note: The final segment of the example has no conditions, and so the results of the highlight query contain all organizations that are linked to the accounts that were results from the penultimate segment. To filter the query results, you can add conditions to the final segment.

When the results of the query are presented to users, they include the values that were exported from the second and third segments:

```
<outputs>
  <field label="# txns Per -- Inter" id="transactioncount"/>
  <field label="# txns Inter -- Org" id="transactioncount2"/>
  <field label="Largest txn value" id="value"/>
  <field label="Most recent txn date" id="dateTime"/>
</outputs>
```

These lines show the challenges of displaying useful information in the relatively confined space of a highlight pane. In fact, only the first two fields appear in the pane; the others are displayed when the user clicks **Show more** to display more results (and more property values from those results). Ideally, the labels work in harmony with the `<description>` of the query that you write to explain the results to your users.

The final part of the highlight query specifies how the application should sort the results. In general, you can request multiple sorts here that are applied in sequence. In this instance, the single criterion is the highest transaction value from the second segment. Setting the `order` to `DESC` means that numbers run from high to low, which is the more common requirement. The opposite is true when you sort on text values, and setting `order` to `ASC` places the results in alphabetical sequence.

Setting up search results filtering

In the clients of an i2 Analyze server, users can filter search results. You can configure the property types and metadata criteria that appear in the filter list by creating and configuring facets for each type of entity and link record that can appear in search results.

About this task

Selecting which property types to use for filtering search results can improve both system performance and the user experience. For example, if the values of properties of a particular type are always unique (license plates, social security numbers), then by default the user sees a separate, one-record facet for every value in the search results. If you configure that property type not to appear, then the system has fewer facets to calculate, and it can display more useful facets from different property types instead.

For records of each entity and link type, the *results configuration file* defines which property types and metadata criteria to use for filtering in Quick Search, external search, and Visual Query results views. All these views display the same set of facets.

The deployment toolkit contains a results configuration file for each of the example schemas in the `examples\schemas` directory, apart from the Chart Store schema. Their names all end in `-results.configuration.xml`. If your system uses a modified version of one of the example schemas, you can modify the appropriate results configuration file.

If you decide to write your own file, the entity and link types that appear must correspond to entity and link types from the deployed schemas. Examine the schemas and the data in your system, and decide which property types to display with facets in the results view. You can also decide which metadata criteria to use for the same purpose.

If you do not specify a results configuration file, all of the property types and metadata criteria that can be displayed with facets, for records of all entity and link types, are displayed in the results view in schema order.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

For more information about the results configuration file and the changes that you can make, see Understanding the results configuration file.

Procedure

1. Using an XML editor, open the results configuration file that you want to modify.
2. Add, modify, or remove any `<ItemTypeFacet>` elements and appropriate child `<PropertyTypeFacet>` and `<MetadataFacet>` elements for your deployment.

   Note: Property types that have the following logical types cannot be used to filter search results:

   - `GEOSPATIAL`
   - `MULTIPLE_LINE_STRING`
3. Save and close the file.

   Note: Ensure that your modified file is stored in the `toolkit\configuration\fragments\common\WEB-INF\classes` directory.
4. If you have created the file or changed its name, you must update the name of the file that the deployment uses.

   a. Using a text editor, open the `DiscoServerSettingsCommon.properties` file in the `toolkit\configuration\fragments\opal-services\WEB-INF\classes` directory.

   b. Ensure that the value of the `ResultsConfigurationResource` property is set to the name of your results configuration file, and then save and close the file.

      Tip: If you do not want to configure search result filtering, clear the value of the `ResultsConfigurationResource` property.

Redeploy i2 Analyze to update the application with your changes.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. Update the i2 Analyze application:

```
setup -t deployLiberty
```

4. Start Liberty:

```
setup -t startLiberty
```

What to do next

Run a selection of queries against an i2 Analyze server. Continue to change the configuration until you are satisfied with the types of filter that are available.

Understanding the results configuration file

In the results configuration file, you specify the schema item types, and their associated property types and metadata criteria, by which users can filter search results. By defining the filtering behavior for different entity and link types, you can refine the filtering to make it as valuable as possible for your users.

The results configuration file has the following basic shape:

```
<!-- Results configuration for Law_Enforcement_Schema.xml -->
<tns:ResultsConfiguration ...>
    <Facets InlineLimit="5" ViewAllLimit="100">
        <!-- Address -->
        <ItemTypeFacet TypeId="ET1" Subfacets="ExcludeSpecific">
            <!-- Unique Reference -->
            <PropertyTypeFacet TypeId="ADD1" />
            ...
            <MetadataFacet Criterion="NotesCreatedBy" />
            ...
        </ItemTypeFacet>
        ...
    </Facets>
</tns:ResultsConfiguration>
```

The `InLineLimit` setting controls the number of values shown for facets in the results grid. Here, the `InLineLimit` is set to the default value of 5. This means that up to 5 values are shown in each facet. You can increase the value, but doing so increases the size of the facets list in the user interface. The `ViewAllLimit` refers to the number of values shown for a facet in the **More** view. The default value for is `ViewAllLimit` is 100. For both settings, higher values can impact performance.

The `<ItemTypeFacet>` element is used to declare which item type you are defining property types and metadata criteria for. The value of the `TypeId` attribute specifies the item type. This value corresponds to the `Id` value of an `<EntityType>` or `<LinkType>` in the schema.

Important: For item types that are defined in gateway or connector schemas, you must augment the item type identifier with the short name of its schema. If the type in the previous example was defined in a gateway schema with the short name `External`, then the `<ItemTypeFacet>` element changes:

```
        <ItemTypeFacet TypeId="ET1-external" Subfacets="ExcludeSpecific">
            ...
        </ItemTypeFacet>
```

In these augmented identifiers, the short name is always in lower-case letters, and any whitespace or non-alphanumeric characters are converted to single hyphens. Furthermore, the short name is always separated from the original identifier with a hyphen.

Property types are specified in child `<PropertyTypeFacet>` elements. The value of the `TypeId` attribute specifies the property type. This value corresponds to the `Id` value of a `<PropertyType>` in the i2 Analyze schema.

Metadata criteria are specified in child <MetadataFacet> elements. The value of the Criterion attribute specifies the metadata criterion and can have the following values:

- **FirstUploaded**

  The date that the record was first uploaded to the Information Store.

- **FirstUploadedBy**

  The display name of the user who first uploaded the record.

- **LastUploaded**

  The date that the record was last uploaded to the Information Store.

- **LastUploadedBy**

  The display name of the user who last uploaded the record.

- **NotesCreatedBy**

  The display name of a user who added a note to the record.

- **SourceRefSourceName**

  The name of a source that appears in the source references for the record.

- **SourceRefSourceType**

  The type of a source that appears in the source references for the record.

- **IngestionDataSourceName**

  The data source name that the record was ingested from.

  If the record was uploaded from Analyst's Notebook, the IngestionDataSourceName is automatically set to ANALYST.

- **StorageLocation**

  For a record that was found through an external search, the location where the record is stored. At this version of the software, the values that users might see are **Information Store** and **Other**.

Note: Any child <MetadataFacet> elements must be specified after all of the child <PropertyTypeFacet> elements within an <ItemTypeFacet> element.

In the <ItemTypeFacet> element, the value of the Subfacets attribute defines the method for specifying the property types and metadata criteria.

The Subfacets attribute can have the following values:

- **All**

  All property types of this item type and metadata criteria are available as filterable options for the results. This behavior is the default if an item type is not added to the results configuration file. For example:

  ```
  <ItemTypeFacet TypeId="ET5" Subfacets="All" />
  ```

  Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' and by all the available properties and metadata.

  Note: You must not specify any child elements when the value of the Subfacets attribute is All.

- **IncludeSpecific**

  Specific property types and metadata criteria for this item type are displayed in the filtering lists. For example:

```
<ItemTypeFacet TypeId="ET5" Subfacets="IncludeSpecific">
    <PropertyTypeFacet TypeId="PER4" />
    <PropertyTypeFacet TypeId="PER6" />
    <MetadataFacet Criterion="NotesCreatedBy" />
</ItemTypeFacet>
```

Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' and by 'First (Given) Name', 'Family Name', and 'NotesCreatedBy'.

Note: You must specify at least one child <PropertyTypeFacet> or <MetadataFacet> element when the value of the Subfacets attribute is IncludeSpecific.

- **ExcludeSpecific**

  Specific property types and metadata criteria for this item type are excluded from the filtering lists. For example:

  ```
  <ItemTypeFacet TypeId="ET3" Subfacets="ExcludeSpecific">
      <PropertyTypeFacet TypeId="VEH2" />
      <MetadataFacet Criterion="FirstUploaded" />
  </ItemTypeFacet>
  ```

  Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Vehicle', but not by 'License Plate Number' or 'FirstUploaded'.

  Note: You must specify at least one child <PropertyTypeFacet> or <MetadataFacet> element when the value of the Subfacets attribute is ExcludeSpecific.

- **None**

  No property types of this item type and metadata criteria are available for filtering. For example:

  ```
  <ItemTypeFacet TypeId="ET5" Subfacets="None" />
  ```

  Declaring this fragment while working with the law enforcement schema will allow you to filter by 'Person' but not by any of the properties of the Person type (such as eye color), nor any of the metadata criteria.

  Note: You must not specify any child elements when the value of the Subfacets attribute is None.

Additionally, you can disable all property type and metadata criteria for faceting by using the PropertyTypeFacetsEnabled and MetadataFacetsEnabled attributes of the <Facets> element. By default, both property type and metadata criteria are enabled for faceting. To disable, set the attribute values to false.

Allowing users to cancel Expand operations

Some Expand operations against the Information Store can find many i2 Analyze records. Returning all of these records to the user might make the chart too crowded or affect system performance. You can configure i2 Analyze to warn users of large result sets and allow them to cancel the operation.

About this task

To specify the number of records that an Expand operation can find before a warning is displayed to analysts, you can provide a value for the ExpandMaxResultsSoftLimit setting in the DiscoServerSettingsCommon.properties file.

If the number of results exceeds this value, Analyst's Notebook displays a message to inform the user. The user can choose to continue or cancel the operation.

By default this setting has a value of 0, which means that no warning is displayed.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Open the `DiscoServerSettingsCommon.properties` file, and provide a value for the `ExpandMaxResultsSoftLimit` setting.

   For example, to display a warning if the number of records exceeds 100:

   ```
   ExpandMaxResultsSoftLimit=100
   ```
2. Save and close the file.

Redeploy i2 Analyze to update the application with your changes.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:

   ```
   setup -t stopLiberty
   ```
3. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```
4. Start Liberty:

   ```
   setup -t startLiberty
   ```

What to do next

After you redeploy i2 Analyze, run a selection of Expand queries to test the behavior of the deployment.

Configuring matching

In a deployment of i2 Analyze that contains data that originates from multiple sources, the likelihood of multiple records that represent the same real-world object or relationship increases. To enable analysts to identify and reduce the number of matching records, you can activate the matching features in Analyst's Notebook by creating custom match rules.

i2 Analyze can be presented with data from multiple sources, through multiple routes. The mechanisms that identify matching records depend on how the data is presented to i2 Analyze:

- For records that are on a chart, *system matching* can compare them with records in the Information Store, and *Find Matching Records* can compare records on the chart with each other.
- When data is presented through Analyst's Notebook import, or retrieved by connectors, or created by analysts, system matching can compare incoming records with records on the chart and in the Information Store.
- During the ingestion process, the incoming records are compared with the records in the Information Store. For more information about identifying matches in this way, see Information Store data correlation.

Important: The scope of match rules is not the same for link records as it is for entity records. Rules for matching entity records apply universally, but rules for matching link records apply only to links between the same pair of ends. Link records that do not connect the same ends are never a match for each other.

This behavior also means that rules for matching link records behave differently in system matching and Find Matching Records:

- In system matching, link records can match only when they are between the same pair of entity *records*.
- In Find Matching Records, link records can match when they are between any two records in a pair of entity *items* on the chart surface.

If the pair of entity items on the chart contain a large number of united records, some combinations of those records are not searched for matching links. By default, the number of link end record combinations that are searched is 100. You can change this value, but the link matching process might take longer if more combinations are searched. For more information about changing the maximum number of combinations that are searched, see The `DiscoServerSettingsCommon.properties` file.

Configuring system matching

System matching uses system match rules to identify matching records. To configure system matching, you define and deploy your system match rules on the i2 Analyze server.

The mechanism that you use to deploy system match rules depends on your deployment pattern:

- In a deployment that contains the Information Store, your system match rules are used to create a match index that determines whether a record matches one that exists in the Information Store. The match index is employed by the Get Matches feature in Analyst's Notebook, and when records are presented through import, connectors, or created by analysts. To create the match index from your match rules, see Deploying system match rules.
- In a deployment that contains only the i2 Connect gateway, a match index is not necessary because the system match rules act only on records in search results and on the chart surface. To deploy your system match rules in this deployment pattern, see Deploying system match rules for the i2 Connect gateway only.

Note: If your deployment does not contain a system match rules file, Analyst's Notebook will still use source identifier matching to determine if records match each other. To change this behavior, review the instructions linked above.

Analysts can choose whether to use system matching at all, but they cannot choose the match rules that are used or approve matches on a record-by-record basis. To ensure that your system match rules do not cause false-positive matches, focus on using known strong identifiers in your data.

If system matching acts on data that involves a large number of seed or matching records (in the tens of thousands), the matching process can take a long time to complete. In this scenario, the following actions might accelerate the process:

- Investigate your match rules to ensure that you are not matching too many records by accident.
- Investigate that the data in your Information Store is being ingested correctly, so that your strong identifiers are not accidentally present on too many records.
- Advise your analysts to use system matching on smaller sets of data.

Deploying system match rules

The system match rules that you deploy are used to create a match index of your data in the Information Store. The match index identifies matches between records that enter the system (or records on a chart) and records in the Information Store.

Before you begin

Ensure that i2 Analyze is started, and that you can connect to the deployment by using Analyst's Notebook.

About this task

In your configuration development environment, use Analyst's Notebook to create your match rules, which are saved to an XML file that you can move to the i2 Analyze server. (The match rules files that Analyst's Notebook creates are compatible with both Find Matching Records and system matching, but i2 recommends that you configure the features separately.)

After you create your match rules file, you create a match index of the data in the Information Store from those rules. In fact, every deployment of i2 Analyze has two match indexes, generated from different versions of the match rules file. One of the indexes is live, and the application uses it to process system match requests. The other index is the standby, which is generated when you modify or write new system match rules.

When the standby index is ready, you switch the standby index to live so that the application uses the match index that reflects your new rules.

Note: If your deployment uses only the i2 Connect gateway, follow the instructions in Deploying system match rules for the i2 Connect gateway only.

Procedure

1. Connect to your server in Analyst's Notebook to load the i2 Analyze schema, and then create your match rules. For more information about how to create match rules, see Find matching records in the Analyst's Notebook documentation. After you create your rules, Analyst's Notebook saves a file named `local-fmr-match-rules.xml` to the `%localappdata%\i2\i2 Analyst's Notebook Premium\Match Rules\<data_source_id>` directory on the local workstation. The file is saved in the directory that was modified most recently.

Next, you must place the match rules file on the i2 Analyze server and update the deployment with your changes.

1. Move the `local-fmr-match-rules.xml` file to the `toolkit\configuration\environment` directory in the i2 Analyze deployment toolkit.
2. Rename `local-fmr-match-rules.xml` to `system-match-rules.xml`.
3. In a command prompt, navigate to the `toolkit\scripts` directory.
4. Update the system match rules on the server and start the population of the standby match index:

   ```
   setup -t updateMatchRules
   ```
5. Monitor the `i2analyze\deploy\wlp\usr\servers\opal-server\logs` directory for a `StandbyMatchIndexReady` file. The file is created when the standby match index completes indexing the data in the Information Store with your new system match rules.
6. Update the application to use the standby match index:

   ```
   setup -t switchStandbyMatchIndexToLive
   ```

This toolkit task switches the standby match index for your new rules to the live index. The match index that was live is now the standby match index.

What to do next

Connect to the deployment again and test your system rules with representative data to ensure that they meet your requirements. In Analyst's Notebook, you must log out and log in again to see your changes.

If you want to modify the system rules again without using Analyst's Notebook, you can modify the file in an XML editor. For more information about the structure of the match rules file, see Match rules syntax.

Deploying system match rules for the i2 Connect gateway only

In a deployment of i2 Analyze that includes only the i2 Connect gateway, the system match rules identify matches between records from external sources and those that are already on a chart. There is no match index, and as a consequence the procedure for deploying system match rules is different from systems with an Information Store.

Before you begin

Ensure that i2 Analyze is started, and that you can connect to the deployment by using Analyst's Notebook.

Also, to reload the server through the reload endpoint, ensure that you have a command-line tool such as `postman` or `curl`, and a user that has the `i2:Administrator` command access control permission.

Note: For more information about using the admin endpoints, see Using the admin endpoints.

About this task

In your configuration development environment, use Analyst's Notebook to create your match rules, which are saved to an XML file that you can move to the i2 Analyze server. (The match rules files that Analyst's Notebook creates are compatible with both Find Matching Records and system matching, but i2 recommends that you configure the features separately.)

Procedure

1. Connect to your server in Analyst's Notebook to load the i2 Analyze schema, and then create your match rules. For more information about how to create match rules, see Find matching records in the Analyst's Notebook documentation. After you create your rules, Analyst's Notebook saves a file named `local-fmr-match-rules.xml` to the `%localappdata%\i2\i2 Analyst's Notebook Premium\Match Rules\<data_source_id>` directory on the local workstation. The file is saved in the directory that was modified most recently.

Next, you must place the match rules file on the i2 Analyze server and update the deployment with your changes.

1. Move the `local-fmr-match-rules.xml` file to the `toolkit\configuration\live` directory in the i2 Analyze deployment toolkit.
2. Delete the `system-match-rules.xml` file from the `toolkit\configuration\live` directory.
3. Rename `local-fmr-match-rules.xml` to `system-match-rules.xml`.

Update the deployment with your changes.

The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

To redeploy your changes by using the deployment toolkit only, see Redeploying Liberty. You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Update the configuration on the server:

   ```
   setup -t updateLiveConfiguration
   ```

   The `updateLiveConfiguration` toolkit task updates server with every file in the `toolkit\configuration\live` directory.
3. Update the running application by using the reload endpoint:

   ```
   curl -i --cookie-jar cookie.txt -d j_username=user_name
                                   -d j_password=password
                                   http://host_name/context_root/
   j_security_check

   curl -i --cookie cookie.txt -X POST http://host_name/context_root/api/v1/
   admin/config/reload
   ```

   To reload the server like this, ensure that you can use the admin endpoints. For more information, see Using the admin endpoints.

   Warning: The `reload` method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

What to do next

Connect to the deployment again and test your system rules with representative data to ensure that they meet your requirements. In Analyst's Notebook, you must log out and log in again to see your changes.

If you want to modify the system rules again without using Analyst's Notebook, you can modify the file in an XML editor. For more information about the structure of the match rules file, see Match rules syntax.

Handling matches in i2 Connect result sets

By default, i2 Analyze returns records from the i2 Connect gateway to clients without reacting to matches among those records. You can edit the `I2ConnectRecordMatchAction` property so that records in the results that match each other arrive on the chart as merged or united records.

You can set `I2ConnectRecordMatchAction` to one of three possible values:

- **IGNORE**

  i2 Analyze ignores any matches between records in the results of an i2 Connect query. This is the default setting.
- **UNITE**

  Records in the results of an i2 Connect query that match each other are added to the chart as entity and link items that contain sets of matching records.
- **MERGE**

  Records in the results of an i2 Connect query that match each other are merged into single records before they reach the chart.

For all three values, and depending on your other settings, the records from the i2 Connect gateway might still match with others when they reach the chart surface.

To change how i2 Analyze handles matching records in i2 Connect result sets:

1. In a text editor, open the `DiscoServerSettingsCommon.properties` file from the `toolkit\configuration\fragments\opal-services\WEB-INF\classes` directory.
2. Find or add the `I2ConnectRecordMatchAction` property and set its value to your preferred setting. For example:

   ```
   I2ConnectRecordMatchAction=UNITE
   ```
3. Save and close the file.
4. In a command prompt, navigate to the `toolkit\scripts` directory and run the following commands to update and restart the Liberty server:

   ```
   setup -t stopLiberty
   setup -t deployLiberty
   setup -t startLiberty
   ```

With the changes in place, you can reconnect to i2 Analyze from i2 Analyst's Notebook and verify that the platform handles matching records in the way that you specified.

Configuring Find Matching Records

Analysts use Find Matching Records to identify when multiple records on an Analyst's Notebook chart might represent the same real-world object or relationship.

Analysts can write their own match rules for Find Matching Records, but you can also configure i2 Analyze to provide a common set of rules to all analysts when they connect. Analysts can choose whether to use all, none, or a selection of the match rules that the server provides. For more information about how analysts might use Find Matching Records, see Creating rules for record matching.

For information about how to create and deploy server-side match rules for Find Matching Records, see Deploying Find Matching Records match rules.

Both server and local match rules for Find Matching Records are tied to a particular server. If a destructive schema change in i2 Analyze takes place, all match rules, whether they are deployed on the server or the client, are invalidated.

Deploying Find Matching Records match rules

In a deployment of i2 Analyze, you can deploy match rules on the server for all Analyst's Notebook users to use in Find Matching Records. The match rules are provided to analysts when they connect to the server.

Before you begin

Ensure that i2 Analyze is started, and that you can connect to the deployment by using Analyst's Notebook.

Also, to reload the server through the reload endpoint, ensure that you have a command-line tool such as `postman` or `curl`, and a user that has the `i2:Administrator` command access control permission.

Note: For more information about using the admin endpoints, see Using the admin endpoints.

About this task

In your configuration development environment, use Analyst's Notebook to create your match rules, which are saved to an XML file that you can move to the i2 Analyze server. (The match rules files that Analyst's Notebook creates are compatible with both Find Matching Records and system matching, but you should configure the features separately.)

Alternatively, you can create the server rules by editing the match rules XML file manually. If you create the match rules this way, you must still complete the following steps that describe how to configure and deploy the file on the i2 Analyze server.

Procedure

1. Connect to your server in Analyst's Notebook to load the i2 Analyze schema, and then create your match rules. As you do so, test them with representative data on the chart.

   For more information about how to create match rules, see Find matching records in the Analyst's Notebook documentation. After you create your rules, Analyst's Notebook saves a file named `local-fmr-match-rules.xml` to the `%localappdata%\i2\i2 Analyst's Notebook \Match Rules\<data_source_id>` directory on the local workstation. The file is saved in the directory that was modified most recently.

   Next, you must place the match rules file on the i2 Analyze server and update the deployment with your changes.
2. Move the `local-fmr-match-rules.xml` file to the `toolkit\configuration\live` directory in the i2 Analyze deployment toolkit.
3. Delete the `fmr-match-rules.xml` file from the `toolkit\configuration\live` directory.
4. Rename `local-fmr-match-rules.xml` to `fmr-match-rules.xml`.

   Update the deployment with your changes.

   The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

   To redeploy your changes by using the deployment toolkit only, see Redeploying Liberty. You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.
5. In a command prompt, navigate to the `toolkit\scripts` directory.
6. Update the configuration on the server:

   ```
   setup -t updateLiveConfiguration
   ```

   The `updateLiveConfiguration` toolkit task updates server with every file in the `toolkit \configuration\live` directory.
7. Update the running application by using the reload endpoint:

   ```
   curl -i --cookie-jar cookie.txt -d j_username=<user_name>
                                   -d j_password=<password>
                                   http://<host_name>/<context_root>/
   j_security_check

   curl -i --cookie cookie.txt -X POST http://<host_name>/<context_root>/api/
   v1/admin/config/reload
   ```

   To reload the server like this, ensure that you can use the admin endpoints. For more information, see Using the admin endpoints.

Warning: The `reload` method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

What to do next

Connect to the deployment again and test that your server rules are visible in the Find Matching Records feature. In Analyst's Notebook, you must log out and log in again to see your changes.

Note: If you copied the `local-fmr-match-rules.xml` file instead of moving it, each rule is duplicated in Analyst's Notebook because the local and server rules are the same.

If you continue to develop your server rules by using this process, the `cached-fmr-match-rules.xml` file in the `Match Rules` directory contains the rules that are currently deployed on the server. Every time that Analyst's Notebook connects, the cached rules file is overwritten with the latest version from the server.

If you want to modify the server rules again without using Analyst's Notebook, you can modify the file in an XML editor. Any changes that you make are validated when you start i2 Analyze. For more information about the structure of the match rules file, see Match rules syntax.

Match rules syntax

A match rules file is an XML document whose structure is validated when i2 Analyze starts. The match rules syntax is the same for both system match and Find Matching Records match rules files.

Root element: matchRules

The root element of a match rules file is a `<matchRules>` element from the defined namespace. For example:

```
<tns:matchRules
  xmlns:tns="http://www.i2group.com/Schemas/2019-05-14/MatchRules"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xsi:schemaLocation=
    "http://www.i2group.com/Schemas/2019-05-14/MatchRules MatchRules.xsd"
  version="2"
  enableSourceIdentifierMatching="true">
  ...
</tns:matchRules>
```

The `<matchRules>` element has the following customizable attributes:

- `enableSourceIdentifierMatching`

  For a deployment that contains the Information Store and the i2 Connect gateway, controls whether system matching uses source identifiers to determine whether records match each other, regardless of whether they have property matches.

  When this attribute is `false` or absent, different users can cause duplication by retrieving the same records from external sources, editing them, and uploading them to the Information Store. When it is `true`, such records are detected by system matching.

- `version`

  The version of the match rules file, which must be 2 at this release.

matchRule

Inside the root element, each <matchRule> element defines a match rule for records of a particular entity or link type. The <matchRule> element has the following attributes:

- id

  An identifier for the match rule, which must be unique within the match rules file.

- itemTypeId

  The identifier for the entity or link type to which the rule applies, as defined in the i2 Analyze schema.

  Important: For item types that are defined in gateway or connector schemas, i2 Analyze appends the schema short name to the item type identifier. For example, if the gateway schema defines an item type with the identifier ET5, then the identifier to use here might be ET5-external.

  In the modified type identifier, the short name is always in lower-case letters and separated from the original identifier with a hyphen. Any whitespace or non-alphanumeric characters in the short name are converted to single hyphens.

  When you create or edit match rules through Analyst's Notebook, the application handles these modifications to item type identifiers for you. When you edit the XML file yourself, you are responsible for specifying item type identifiers correctly.

- displayName

  The name of the rule, which is displayed to analysts in Analyst's Notebook in Find Matching Records.

- description

  The description of the rule, which is displayed to analysts in Analyst's Notebook in Find Matching Records.

- active

  Defines whether the rule is active. A value of true means that the rule is active; a value of false means that the rule is not active.

- linkDirectionOperator

  Determines whether two links must have the same direction in order to match. Mandatory for link type rules, where it must have the value EXACT_MATCH or ANY. Must be absent or null for entity type rules.

- version

  In earlier versions, the version attribute was mandatory on <matchRule> elements. At this release, the per-rule version is optional, and any value is ignored.

For example, an entity type rule:

```
<matchRule
  id="4a2b9baa-e3c4-4840-a9fd-d204711af50e"
  itemTypeId="ET3"
  displayName="Match vehicles"
  description="Match vehicles with the same license plate when
               either the registered state or region are the same."
  active="true">
  ...
</matchRule>
```

And a link type rule:

```
<matchRule
```

```
  id="8aa5f6f4-a1a8-41de-b5c5-8701e44bcde7"
  itemTypeId="LAS1"
  displayName="Match duplicate links"
  description="Match link records between the same pair of entity
               records when the links are in the same direction."
  active="true"
  linkDirectionOperator="EXACT_MATCH">
  ...
</matchRule>
```

matchAll and matchAny

To specify the behavior of a match rule, you can use the following children of the `<matchRule>` element:

`matchAll`

The `<matchAll>` element specifies that all of the conditions within it must be met.

`matchAny`

The `<matchAny>` element specifies that at least one of the conditions within it must be met.

A `<matchRule>` element must have both the `<matchAll>` and `<matchAny>` elements. For example:

```
<matchRule ... >
  <matchAll>
    ...
  </matchAll>
  <matchAny />
</matchRule>
```

condition

All match rules must contain the `<matchAll>` and `<matchAny>` elements, although both can be empty for link type rules. It is valid to create a rule that makes all link records of the same type between the same pair of entity records match each other, regardless of any other considerations.

All entity type rules must contain at least one condition. Many link type rules contain conditions too. Each condition defines a comparison that takes place between values in different records, and specifies when those values are considered to match. Conditions can be refined by using operators, values, and normalizations.

To specify the conditions of a match rule, you use the `<condition>` element that can be a child of the `<matchAll>` and `<matchAny>` elements. Each `<condition>` element has a mandatory `propertyTypeId` attribute, which is the identifier for the property type to which the condition applies, as defined in the i2 Analyze schema.

For example:

```
<condition propertyTypeId="VEH2">
...
</condition>
```

All conditions contain an `<operator>` element, most of them a contain `<value>` element, and many contain `<normalizations>`.

`operator`

The `<operator>` element defines the type of comparison between the property values in different records, or between the property value and a static value specified within the rule. The possible operators are:

Operator	Description
EXACT_MATCH	The values that are compared must match each other exactly.
EXACT_MATCH_START	A specified number of characters at the start of string values must match each other exactly.
EXACT_MATCH_END	A specified number of characters at the end of string values must match each other exactly.
EQUAL_TO	The property values must match each other, and the specified `<value>`.

For example:

```
<condition propertyTypeId="VEH2">
  <operator>EXACT_MATCH</operator>
  ...
</condition>
```

`value`

The contents of the `<value>` element affect the behavior of the `<operator>` of the condition. Different operators require different value types.

- If the operator is `EXACT_MATCH_START` or `EXACT_MATCH_END`, the value is an integer that specifies the number of characters to compare at the start or end of the property value:

  ```
  <operator>EXACT_MATCH_START</operator>
  <value xsi:type="xsd:int">3</value>
  ```

- If the operator is `EQUAL_TO`, the value is a string to compare with the property value:

  ```
  <operator>EQUAL_TO</operator>
  <value xsi:type="xsd:string">red</value>
  ```

- If the operator is `EXACT_MATCH`, it is not valid to specify a `<value>` element.

  ```
  <operator>EXACT_MATCH</operator>
  ```

`normalizations`

The `<normalizations>` element contains child `<normalization>` elements that define how property values are compared with each other (and sometimes with the contents of the `<value>` element). The possible values for the `<normalization>` element are:

Normalization	Description
IGNORE_CASE	Ignores case during the comparison (`'a'` matches `'A'`)
IGNORE_DIACRITICS	Ignores diacritic marks on characters (`'Ã'` matches `'A'`)
IGNORE_WHITESPACE_BETWEEN	Ignores whitespace between characters (`'a a'` matches `'aa'`)

Normalization	**Description**
IGNORE_WHITESPACE_AROUND	Ignore whitespace around a string (`' a '` matches `'a'`)
IGNORE_NUMERIC	Ignore numeric characters (`'a50'` matches `'a'`)
IGNORE_ALPHABETIC	Ignore alphabetic characters (`'a50'` matches `'50'`)
IGNORE_NONALPHANUMERIC	Ignore non-alphanumeric characters (`'a-a'` matches `'aa'`)
SIMPLIFY_LIGATURES	Simplify ligatures (`'æ'` matches `'ae'`)

For example, you might have the following normalizations for an `EXACT_MATCH` operator:

```
<condition ... >
  <operator>EXACT_MATCH</operator>
  <normalizations>
    <normalization>IGNORE_CASE</normalization>
    <normalization>IGNORE_NONALPHANUMERIC</normalization>
    <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
  </normalizations>
</condition>
```

In this example, the values "b m w xdrive" and "BMW x-drive" are considered a match.

The operators and normalizations that you can specify for a condition depend on the logical type of the property type to which the condition applies. The following table shows the operators and normalizations that you can use for each logical type:

Schema logical type	**Operators**	**Normalization**
SINGLE_LINE_STRING	All	All
SELECTED_FROM	All	All
SUGGESTED_FROM	All	All
BOOLEAN	EXACT_MATCH	None
INTEGER	EXACT_MATCH	None
DECIMAL	EXACT_MATCH	None
DOUBLE	EXACT_MATCH	None
DATE_AND_TIME	EXACT_MATCH	None
DATE	EXACT_MATCH	None

Schema logical type	Operators	Normalization
TIME	EXACT_MATCH	None

Property types that have the following logical types cannot be used in match rules:

- GEOSPATIAL
- MULTIPLE_LINE_STRING

The following XML is an example of a match rules file that contains a single entity match rule. The rule matches vehicle records that have the same values for the license plate property, and the same values for either the state or region properties.

For example, two vehicle records with the license plates "1233 DC 33" and "1233DC33" from the regions "Ile-de-France and "Île De France" are identified as a match for the following rule:

```
<tns:matchRules
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
      "http://www.i2group.com/Schemas/2019-05-14/MatchRules MatchRules.xsd"
    version="2"
    xmlns:tns="http://www.i2group.com/Schemas/2019-05-14/MatchRules">
  <matchRule id="4a2b9baa-e3c4-4840-a9fd-d204711af50e"
             itemTypeId="ET3"
             displayName="Match vehicles"
             description="Match vehicles with the same license plate,
                          when either the registered state or
                          region are the same."
             active="true">
    <matchAll>
      <condition propertyTypeId="VEH2">
        <operator>EXACT_MATCH</operator>
        <normalizations>
          <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
        </normalizations>
      </condition>
    </matchAll>
    <matchAny>
      <condition propertyTypeId="VEH16">
        <operator>EXACT_MATCH</operator>
        <normalizations>
          <normalization>IGNORE_CASE</normalization>
          <normalization>IGNORE_DIACRITICS</normalization>
          <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
          <normalization>IGNORE_NONALPHANUMERIC</normalization>
        </normalizations>
      </condition>
      <condition propertyTypeId="VEH15">
        <operator>EXACT_MATCH</operator>
        <normalizations>
          <normalization>IGNORE_CASE</normalization>
          <normalization>IGNORE_DIACRITICS</normalization>
          <normalization>IGNORE_WHITESPACE_BETWEEN</normalization>
          <normalization>IGNORE_NONALPHANUMERIC</normalization>
        </normalizations>
      </condition>
    </matchAny>
  </matchRule>
```

```
</tns:matchRules>
```

Configuring geospatial mapping

Analyst's Notebook supports several different mechanisms for mapping chart items, but only one that supports geographic data that is stored in i2 Analyze records. You must define the maps and geographic reference systems to use before the mapping features that can be used with i2 Analyze records can be enabled.

When records contain geographic information, Analysts can view their location on supplied maps. In addition, Analysts can use visual queries to search for information in specific locations.

The mapping configuration file contains the following main types of information:

Mapping tiles

The mapping configuration file contains information that identifies the map tiles that are available to Analysts. The following details are needed for each map:

- A `displayName` for the mapping tile to display to the Analyst.
- The URL that hosts the mapping tile.
- Any information about the origins of the mapping tile, such as the source and any copyright details.

Base Maps

Base maps form the background on which other GIS items are overlaid. Depending on the type of investigation, and the location, different types of base map might be suitable. For example, you might want to use graphical representations or satellite imagery. In addition to the details that are needed for all mapping tiles, you can specify which base map is opened by default.

Overlays

Layers that sit over the base map and contain detailed information that might be relevant to specific types of investigation. For example, cell tower locations, buildings of interest, borders, or road systems. By adding such information to layers that are overlaid over a base map, you can select the relevant information without displaying information that isn't needed. Unlike base maps, as different overlays can contain different information, multiple overlay tiles can be added to the mapping view. As overlay-mapping tiles are added over other maps, you can set the opacity level to see information on lower mapping tiles.

Coordinate Systems

The 2D coordinate reference systems that can be used to translate coordinates to map locations. The following details are needed for each coordinate system:

- A `displayName` for the coordinate system to display to the Analyst.
- The projection string that defines the coordinate system. This string must be a valid PROJ4 coordinate reference system string.
- The type of editor to use to input the values.

In addition, you can define the bounds to use with the coordinates to constrain the area being mapped.

Defining the mapping solution

In a deployment of i2® Analyze, you can define the mapping tiles that can be accessed, and the coordinate systems that are used to plot locations on them. By default, analysts can select from a list of publicly available maps and common coordinate systems.

About this task

Analysts can access mapping features to help investigate geographic information. You might need to update the mapping tiles that are provided, or the coordinate reference systems that are used to calculate locations.

Procedure

1. Modify the `geospatial-configuration.json` file:
   a. In your JSON editor, open the `toolkit\configuration\live\geospatial-configuration.json` file.
   b. Using the reference and example information, modify the file to define the mapping tiles and coordinate reference systems your analysts can use.
2. Update the deployment with your changes.

   The following method deploys your changes without stopping the server by using a POST request to a REST endpoint.

   Note: To redeploy your changes by using the deployment toolkit only, see Redeploying Liberty. You must deploy your changes by using the deployment toolkit if you are in a deployment with high availability or you are deploying in your production environment.

   a. Update the configuration on the server:

   ```
   setup -t updateLiveConfiguration
   ```

   b. Update the running application by using the reload endpoint:

   ```
   curl -i --cookie-jar cookie.txt -d j_username=user_name -d
    j_password=<password> http://<host_name>/<context_root>/
   j_security_check

   curl -i --cookie cookie.txt -X POST http://<host_name>/<context_root>/
   api/v1/admin/config/reload
   ```

   Note: To reload the server by using the reload endpoint, ensure that you can use the admin endpoints. For more information about using the admin endpoints, see Using the admin endpoints.

   Warning: The `reload` method updates the configuration without requiring a server restart, but any logged-in users are logged out from i2 Analyze when you run it.

Results

Your configuration is validated, and any errors are reported in the REST response and in the `wlp\usr\servers\opal-server\logs\<deployed_war>\i2_Update_Live_Configuration.log`.

Where `<deployed_war>` can be:

- `opal-services-is`
- `opal-services-daod`
- `opal-services-is-daod`

What to do next

Depending on the type of spatial data that you are ingesting into your Information Store, to increase the performance of Visual Query operations, you might need to set up spatial database indexes on `IS_DATA` tables that contain geographical data. For more information on creating indexes manually in the Information Store, see Creating indexes in the Information Store database. In addition, for database-specific information about the use of spatial indexes:

- For Db2 databases, see the IBM Integrated Analytics System documentation: Using indexes and views to access spatial data.
- For SQL Server databases, see the Microsoft SQL Server Spatial Indexing documentation: https://learn.microsoft.com/sql/relational-databases/spatial/spatial-indexes-overview.
- For PostgreSQL databases, see the PostGIS Spatial Indexing documentation: https://postgis.net/workshops/postgis-intro/indexing.html.

The example mapping configuration file

The toolkit `information-store-opal` and `information-store-daod-opal` example configurations contain an example `geospatial-configuration.json` file in the `configuration/live` folder. This example defines publicly available mapping tiles, and common coordinate reference systems.

The `geospatial-configuration.json` file that is included in the deployment toolkit contains:

Base Map - Esri World Street Map

A link to the publicly available Esri World Street Map tile that presents highway-level data for the world and street-level data for specific regions such as North America, Europe, and Africa. For more information about the Esri World Street Map, see https://www.arcgis.com/home/item.html?id=3b93337983e9436f8db950e38a8629af.

Coordinate Systems

The following coordinate systems are included:

- **OSGB36 National Grid**

  Used in Great Britain.
- **NAD83 / UTM zone 10N**

  Used in North America - between 126°W and 120°W - onshore and offshore. Canada - British Columbia; Northwest Territories; Yukon. United States (USA) - California; Oregon; Washington.
- **NAD83 / BC Albers**

  Used in Canada - British Columbia.
- **WGS 84 / UPS North (N,E)**

  Used in the northern hemisphere - North of 60°N onshore and off shore, including Arctic.
- **WGS 84 / UPS South (N,E)**

  Southern hemisphere - south of 60°S onshore and offshore - Antarctica.
- **WGS 84 / Pseudo-Mercator**

  Used in World between 85.06°S and 85.06°N.

You might need to update this file for a number of reasons, for example:

- Network access - If you are within a network that is not connected to the internet, you need to host mapping tiles on an internal server, and provide details of your internally hosted mapping tiles. In this scenario, as the example maps are not accessible, the original maps should also be removed.
- Additional Maps - You can add references to additional mapping tiles that are either publicly available or hosted locally. This allows you to add different base map tiles to include features such as satellite imagery, and different overlay layers to include items of interest such as key transportation routes or buildings of a particular type.
- Handling alternative coordinate systems - If you are aware that coordinates in your system use a particular coordinate reference system, you can update the list to include a different coordinate system that covers a specific area such as the Alberta 3-TM transformation. Alternatively, you can remove coordinate reference systems that are unlikely to be used in your deployment.

The example `geospatial-configuration.json` file:

```
{
  "mapConfig": {
    "baseMaps": [
      {
        "id": "ESRI_WorldStreetMap",
        "displayName": "ESRI World Street Map",
        "url": "https://server.arcgisonline.com/ArcGIS/rest/services/
World_Street_Map/MapServer/tile/{z}/{y}/{x}",
        "attribution": "Sources: Esri, HERE, Garmin, USGS, Intermap,
 INCREMENT P, NRCan, Esri Japan, METI, Esri China (Hong Kong), Esri Korea,
 Esri (Thailand), NGCC, ? OpenStreetMap contributors, and the GIS User
 Community"
      }
    ]
  },
  "coordinateSystems": [
    {
      "id": "EPSG:27700",
      "displayName": "OSGB 1936 / British National Grid - United Kingdom
 Ordnance Survey",
      "projString": "+proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717
 +x_0=400000 +y_0=-100000 +ellps=airy +datum=OSGB36 +units=m +no_defs",
      "editorType": "EASTING_NORTHING",
      "bounds": {
        "north": 1300000,
        "east": 700000,
        "south": 0,
        "west": 0
      }
    },
    {
      "id": "EPSG:26910",
      "displayName": "NAD83 / UTM zone 10N",
      "projString": "+proj=utm +zone=10 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0
 +units=m +no_defs",
      "editorType": "EASTING_NORTHING"
    },
    {
      "id": "EPSG:3005",
      "displayName": "NAD83 / BC Albers",
      "projString": "+proj=aea +lat_1=50 +lat_2=58.5 +lat_0=45 +lon_0=-126
 +x_0=1000000 +y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs",
      "editorType": "EASTING_NORTHING"
```

```
    },
    {
      "id": "EPSG:32661",
      "displayName": "WGS 84 / UPS North (N,E)",
      "projString": "proj4.defs(\"EPSG:32661\",\"+proj=stere +lat_0=90
 +lat_ts=90 +lon_0=0 +k=0.994 +x_0=2000000 +y_0=2000000 +datum=WGS84
 +units=m +no_defs\");",
      "editorType": "EASTING_NORTHING"
    },
    {
      "id": "EPSG:32761",
      "displayName": "WGS 84 / UPS South (N,E)",
      "projString": "+proj=stere +lat_0=-90 +lat_ts=-90 +lon_0=0 +k=0.994
 +x_0=2000000 +y_0=2000000 +datum=WGS84 +units=m +no_defs",
      "editorType": "EASTING_NORTHING"
    },
    {
      "id": "EPSG:3857",
      "displayName": "WGS 84 / Pseudo-Mercator -- Spherical Mercator, Google
 Maps, OpenStreetMap, Bing, ArcGIS, ESRI",
      "projString": "+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0
 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +wktext  +no_defs",
      "editorType": "EASTING_NORTHING"
    }
  ]
}
```

The `geospatial-configuration.json` file

The `geospatial-configuration.json` file contains definitions of the mapping tiles and coordinate reference systems to use in your deployment. You can use this reference and example information when you create your own mapping configuration file.

Whether you choose to update the example provided, or create your own file, the `geospatial-configuration.json` contains the following key elements:

- `mapConfig`
  - `baseMaps`
  - `overlays`
- `coordinateSystems`

`mapConfig`

The `mapConfig` object defines the mapping tiles that are available.

The `mapConfig` object can contain a single instance of each of the following objects that are used to define the mapping view and to provide defaults for all mapping layers that are defined in the file.

- `center` (Type: *Lat/Long*; Default: *Undefined*)

  The geographic center of the map when it is first opened.

  The `lat` and `lng` fields are used to specify the center point.
- `zoom` (Type: *Number*; Default: *Undefined*)

  The map zoom level when it is first opened.
- `maxBounds` (Type: *Lat/Long Bounds*; Default: *Undefined*)

The maximum area that can be mapped by defining the coordinates of two diagonally opposite corners of a rectangle.

- `minZoom` (Type: *Number*; Default: *none*)

  The minimum amount the map can zoom (inclusive). If this option is not specified, the minimum zoom level is calculated as the highest minimum zoom option available in any defined `GridLayer` or `TileLayer`.

- `maxZoom` (Type: *Number*; Default: *none*)

  The maximum amount the map can zoom (inclusive). If this option is not specified, the maximum zoom level is calculated as the lowest maximum zoom option available in any defined `GridLayer` or `TileLayer`.

- `wrap` (Type: *Boolean*; Default: *True*)

  Whether any specified layers are wrapped around the antimeridian. If this option is set to `false`, the layer is only displayed once at low zoom levels.

Examples:

```
"mapConfig": {
  "center": {
    "lat": 52.202468,
    "lng": 0.142655
  },
  "zoom": 8,
  "maxBounds": {
    "north": 60.84,
    "south": 49.96,
    "east": 1.78,
    "west": -7.56
  },
  "minZoom": 2,
  "maxZoom": 10,
  "wrap": false,
...
}
```

In addition, depending on your mapping requirements, it must contain one or more base maps and can contain one or more overlays

`baseMaps`

A base map is an image that forms a background over which other GIS items are overlaid. All base maps must be hosted externally, and can be referenced by using a URL. If you would like to reference more than one base map, you can set a default that is used when a map is requested.

- `id` (Type: *Text*; Default: *none*)

  The unique identifier used to distinguish the layer. (Mandatory)

- `displayName` (Type: *Text*; Default: *none*)

  The name to use to refer to this map when the map is displayed. (Mandatory)

- `url` (Type: *String*; Default: *none*)

  The URL template for the tile server that hosts the mapping image. This URL can be a path to a standard image file format: `png`, or `jpg`, or a tile layer. (Mandatory)

  The URL template can contain the following variables:

- `{x}` and `{y}` - the tile coordinates
- `{z}` - the zoom level
- `{s}` - substituted with the list of subdomains, a separate request is made for each subdomain
- `{r}` - add "@2x" to load retina tiles

Note: i2 Maps use tiles that are based on the EPSG:3857 projected coordinate system. Using an unsupported map type does not throw an error, but the map is not rendered correctly.

- `attribution` (Type: *String*; Default: *none*)

  Information about the origin of the map that is displayed in the lower right of the map.

  Hyperlinks are supported alongside other text. You can identify hyperlinks by tagging them in the following format: `[hyperlink]url[/hyperlink]`.

- `tilesize` (Type: *Number*; Default: *265*)

  The tile size in pixels.

- `minZoom` (Type: *Number*; Default: *none*)

  The minimum amount the map can zoom (inclusive).

  If this option is not specified, the minimum zoom level is calculated as the highest minimum zoom option available in the layer.

- `maxZoom` (Type: *Number*; Default: *none*)

  The maximum amount the map can zoom (inclusive).

  If this option is not specified, the maximum zoom level is calculated as the lowest maximum zoom option available in the layer.

- `subdomains` (Type: *String array*; Default: *none*)

  Subdomains of the tile service. Can be passed in the form of a string array.

- `crossOrigin` (Type: *String*; Default: *Anonymous*)

  Defines how the tile server is configured to handle crossorigin requests. This can be one of the following values:

  - `anonymous`
  - `use-credentials`
  - `""`

- `tileBounds` (Type: *Lat/Long Bounds*; Default: *none*)

  The WGS84 bounds for the tiles, allowing you to constrain the tile loading area to a specific location. For example:

  ```
  tileBounds: {
    west: 0.72235,
    east: 0.79101,
    south: 52.20424,
    north: 52.35211
  }
  ```

- `defaultBaseMap` (Type: *Boolean*; Default: *False*)

  If there are multiple base maps, this option identifies the map to load initially.

Example:

```
"baseMaps": [
  {
    "id": "Esri.DeLorme",
    "displayName": "Esri Delorme",
    "url": "https://server.arcgisonline.com/ArcGIS/rest/services/Specialty/
DeLorme_World_Base_Map/MapServer/tile/{z}/{y}/{x}",
    "attribution": "Tiles © Esri — Copyright: © 2012 DeLorme"
  }
]
```

`overlays`

An overlay is a type of map layer that is designed to provide additional information in addition to the base map. As these images for a secondary layer over the base map, you can set the opacity of an overlay to allow information present in the base map to be displayed.

- `id` (Type: *Text*; Default: *none*)

  The unique identifier used to distinguish the layer. (Mandatory)

- `displayName` (Type: *Text*; Default: *none*)

  The name to use to refer to this map when the map is displayed. (Mandatory)

- `url` (Type: *String*; Default: *none*)

  The URL template for the tile server that hosts the mapping image. This URL can be a path to a standard image file format: `png`, or `jpg`, or a tile layer. (Mandatory)

  The URL template can contain the following variables:

  - `{x}` and `{y}` - the tile coordinates
  - `{z}` - the zoom level
  - `{s}` - substituted with the list of subdomains, a separate request is made for each subdomain
  - `{r}` - add "@2x" to load retina tiles

  Note: i2 Maps use tiles that are based on the EPSG:3857 projected coordinate system. Using an unsupported map type does not throw an error, but the map is not rendered correctly.

- `attribution` (Type: *String*; Default: *none*)

  Information about the origin of the map that is displayed in the lower right of the map.

  Hyperlinks are supported alongside other text. You can identify hyperlinks by tagging them in the following format: `[hyperlink]url[/hyperlink]`.

- `tilesize` (Type: *Number*; Default: *265*)

  The tile size in pixels.

- `minZoom` (Type: *Number*; Default: *none*)

  The minimum amount the map can zoom (inclusive).

  If this option is not specified, the minimum zoom level is calculated as the highest minimum zoom option available in the layer.

- `maxZoom` (Type: *Number*; Default: *none*)

  The maximum amount the map can zoom (inclusive).

  If this option is not specified, the maximum zoom level is calculated as the lowest maximum zoom option available in the layer.

- `subdomains` (Type: *String array*; Default: *none*)

  Subdomains of the tile service. Can be passed in the form of a string array.

- `crossOrigin` (Type: *String*; Default: *Anonymous*)

  Defines how the tile server is configured to handle crossorigin requests. This can be one of the following values:

  - `anonymous`
  - `use-credentials`
  - `""`

- `tileBounds` (Type: *Lat/Long Bounds*; Default: *none*)

  The WGS84 bounds for the tiles, allowing you to constrain the tile loading area to a specific location. For example:

  ```
  tileBounds: {
    west: 0.72235,
    east: 0.79101,
    south: 52.20424,
    north: 52.35211
  }
  ```

- `opacity` (Type: *Number*; Default: *1.0*)

  The degree of visibility for objects on this layer. If set, the opacity must be a decimal between 0.0 (not visible) and 1.0 (fully visible).

Example:

```
"overlays": [
  {
    "id": "OpenMapSurfer_AdminBounds",
    "displayName": "OpenMapSurfer Admin Bounds",
    "url": "https://maps.heigit.org/openmapsurfer/tiles/adminb/webmercator/
{z}/{x}/{y}.png",
    "attribution": "Imagery from [hyperlink]http://giscience.uni-hd.de/
 GIScience Research Group @ University of Heidelberg[/hyperlink] | Map
 data © [hyperlink]https://www.openstreetmap.org/copyright OpenStreetMap[/
hyperlink] contributors"
  }
]
```

`coordinateSystems`

The `coordinateSystems` object defines the coordinate reference systems that are available.

- `id` (Type: *Text*)

  The unique identifier used to distinguish the coordinate system.

- `displayName` (Type: *Text*)

  The name to use to refer to this coordinate system when displayed in the UI.

- `projString` (Type: *Text*)

  The projection string used to define the coordinate system. This string is a CRS definition.

- `editorType` (Type: *Text*)

  The controls to be used to label the x and y axis. This type can be:

- `LATITUDE_LONGITUDE` - Ellipsoidal 2D CS
- `EASTING_NORTHING` - Cartesian 2d CS E,N
- `X_Y` - Cartesian 2d CS X,Y

- `bounds` (Type: *Lat/Long Bounds*)

  The area that can be mapped by defining the coordinates of two diagonally opposite corners of a rectangle. (Optional)

Example:

```
"coordinateSystems": [
  {
    "id": "EPSG:3081",
    "displayName": "NAD83 / Texas State Mapping System",
    "projString": "+proj=lcc +lat_1=27.41666666666667
 +lat_2=34.91666666666666 +lat_0=31.16666666666667 +lon_0=-100 +x_0=1000000
 +y_0=1000000 +ellps=GRS80 +datum=NAD83 +units=m +no_defs",
    "editorType": "X_Y"
  }
]
```

Configuring source references

A source reference provides information about a source that some or all of the data for a record or chart came from. Source references are displayed to analysts in Analyst's Notebook, i2 Notebook, and the Investigate Add On.

You can specify the contents of source references during ingestion, retrieval of data from external sources. Analysts can also create source references in Analyst's Notebook.

A source reference consists of the following fields:

Source name

The name of the source. This is the only field that must contain a value.

For example, "Phone Network".

Source type

The type of the source.

For example, "Cell data".

Source location

The location of the source. This might be a description of a physical location, or a URL to a digital location. If the location value is in a URL format, it is clickable by analysts.

For example, "Filing cabinet 32, drawer 4" or "http://www.exampleData.com/phone1.html".

Source image URL

A URL to an image of the source. If the URL resolves to an image, it is displayed with the record in Analyst's Notebook and the Investigate Add On.

For example, "http://www.exampleImages.com/phone1.jpg".

Source description

A description of the source.

For example, "This source contains highly accurate information about cell data from the phone network provider".

Specifying the contents of source references

Source references are created or provided in different parts of the system, and you configure the possible values in different ways depending on how the source reference enters the system.

Ingestion source references

The values for source reference name and description are taken from the ingestion source name and description that is specified in the mapping file that is used when you ingest the data.

Records acquire a source reference during the ingestion process. You can provide values for the source reference type, location, and image URL fields for each row in the staging table when you ingest data into the Information Store.

For more information about providing values for this type of source reference, see Information Store staging tables.

Analyst-created source references

Analysts can add source references to records and charts by using Analyst's Notebook. You can constrain the values that analysts can use for the source reference name and type fields.

For more information about configuring these values, see Configuring analyst-created source references.

Connector source references

When records are retrieved from an external data source via the i2 Connect gateway, the connector implementation can provide a source reference.

For more information about implementing source references into your connectors, see Returning source references.

Configuring analyst-created source references

Analysts can create source references on i2 Analyze records and charts by using the **Record Inspector** or **Chart Inspector** in Analyst's Notebook. By default, analysts can use any values for the source reference name and type, but you can restrict the values that are available for them to use.

Before you begin

It is recommended that you configure analyst-created source references in your configuration development environment before you implement them in your production environment.

About this task

You can restrict the values that analysts can specify for the *source name* and *source type* for each item type and for charts. Analysts can provide single-line string values, or you can define selected-from or suggested-from lists that they can choose from. You might choose to restrict the values that analysts can specify to improve the quality of the source references in a system. By enforcing selected-from lists, you can ensure consistent names within your system.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. Enable the use of the source reference schema.
   a. Create your `source-reference-schema.xml` file XML schema file in the `configuration\fragments\opal-services\WEB-INF\classes` directory.
   b. Specify your source reference schema file name as the value for the `SourceReferenceSchemaResource` property in the `configuration\fragments\opal-services\WEB-INF\classes\DiscoServerSettingsCommon.properties` file.

      For example:

      ```
      SourceReferenceSchemaResource=source-reference-schema.xml
      ```
2. Modify the source reference schema configuration file.
   a. Open your XSD aware XML editor. For more information see Setting up your XSD aware XML editor. The associated XSD file is: `toolkit\scripts\xsd\SourceReferenceSchema.xsd`.
   b. In your XML editor, open your `source-reference-schema.xml` file.
   c. Using the reference and example information, modify the file to define the values that analysts can use.
3. Update the deployment with your changes. On the Liberty server, in a command prompt navigate to the `toolkit\scripts` directory:

   ```
   setup -t deployLiberty
   setup -t startLiberty
   ```
4. Test the source reference schema that you created.
   a. In Analyst's Notebook, use the **Record Inspector** or **Chart Inspector** to create source references on records and charts. In Analyst's Notebook, you must log out and log in again to see your changes.
   b. Ensure that the values that analysts can use meet your requirements.

What to do next

Complete this process as many times as necessary until the source reference schema meets your requirements.

The `source-reference-schema.xml` file

The source reference schema XML file has the following structure. You can use this reference and example information when you create your own source reference schema file.

- `<sourceReferenceSchema>`
- `<sourceReferenceSchemaFragments>`, `<sourceReferenceSchemaFragment>`
- `<sourceName>`
- `<sourceType>`
- `<possibleValues>`, `<possibleValue>`

`<sourceReferenceSchema>`

The `<sourceReferenceSchema>` element is the root of the configuration file.

`<sourceReferenceSchemaFragments>`, `<sourceReferenceSchemaFragment>`

The <sourceReferenceSchemaFragments> element can contain one or more <sourceReferenceSchemaFragment> child elements.

The <sourceReferenceSchemaFragment> defines the values that analysts can use for the source name and source type for a particular set of item types.

The itemTypeIds attribute specifies the item types within a set. You must use the item type IDs that come from the i2 Analyze schema for the itemTypeIds attribute. You can specify multiple item type IDs as a comma-separated list. If you do not specify any item type IDs, the restrictions apply to all item types that are not referenced elsewhere in the source reference schema.

For example, to make a set for the source references associated with charts and the item type with ID "ET5":

```
<sourceReferenceSchemaFragments>
  <sourceReferenceSchemaFragment itemTypeIds="CHART,ET5">
    ...
  </sourceReferenceSchemaFragments>
```

The <sourceReferenceSchemaFragment> can contain one child <sourceName> element and one child <sourceType> element.

<sourceName>

The <sourceName> element contains the possible values that analysts can use as a source name.

The logicalType attribute defines whether analysts must choose a value or can provide their own values. The values for the logicalType attribute are:

- SELECTED_FROM - You provide a list of source names that analysts can use.
- SUGGESTED_FROM - You provide a list of source names that analysts can use and they can create their own.
- SINGLE_LINE_STRING - Analysts create their own source names.

If you specify the SELECTED_FROM or SUGGESTED_FROM logical types, use the child <possibleValues> element to specify the values that analysts can use.

<sourceType>

The <sourceType> element contains the possible values that analysts can use as a source type.

The logicalType attribute defines whether analysts must choose a value or can provide their own values. The values for the logicalType attribute are:

- SELECTED_FROM - You provide a list of source types that analysts can use.
- SUGGESTED_FROM - You provide a list of source types that analysts can use and they can create their own.
- SINGLE_LINE_STRING - Analysts create their own source types.

The mandatory attribute defines whether analysts must specify a value for the source type when they create a source reference. The possible values are true and false.

If you specify the SELECTED_FROM or SUGGESTED_FROM logical types, use the child <possibleValues> element to specify the values that analysts can use.

<possibleValues>, <possibleValue>

The <possibleValues> element contains child <possibleValue> elements. You can specify the <possibleValues> element when its parent element specifies SELECTED_FROM or SUGGESTED_FROM for the logicalType attribute.

The `value` attribute of the `<possibleValue>` element contains the value that is displayed to analysts to select.

Example

In this example, analysts can create source references for charts with the source name "Local Police Department" or "Analyst Team 1" to describe the source of the chart.

For item type ET5, for the source name, analysts can use one of the three possible values from the selected-from list. For the source type, they can use one of the three values in the suggested-from list, or provide their own value.

For all other item types, analysts can use any values for the name and type.

```
<tns:sourceReferenceSchema
    xmlns:tns="http://www.i2group.com/Schemas/2019-07-05/
SourceReferenceSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.i2group.com/Schemas/2019-07-05/
SourceReferenceSchema SourceReferenceSchema.xsd">
  <sourceReferenceSchemaFragments>

    <sourceReferenceSchemaFragment itemTypeIds="CHART" >
      <sourceName logicalType="SELECTED_FROM">
        <possibleValues>
          <possibleValue value="Local Police Department"/>
          <possibleValue value="Analyst Team 1"/>
        </possibleValues>
      </sourceName>
      <sourceType logicalType="SINGLE_LINE_STRING" mandatory="false">
      </sourceType>
    </sourceReferenceSchemaFragment>

    <sourceReferenceSchemaFragment itemTypeIds="ET5" >
      <sourceName logicalType="SELECTED_FROM">
        <possibleValues>
          <possibleValue value="Police Department"/>
          <possibleValue value="Phone Network"/>
          <possibleValue value="Social Media"/>
        </possibleValues>
      </sourceName>

      <sourceType logicalType="SUGGESTED_FROM" mandatory="true">
        <possibleValues>
          <possibleValue value="Crime Report"/>
          <possibleValue value="RFI - Cell Data"/>
          <possibleValue value="RFI - Social Media"/>
        </possibleValues>
      </sourceType>
    </sourceReferenceSchemaFragment>

    <sourceReferenceSchemaFragment>
      <sourceName logicalType="SINGLE_LINE_STRING">
      </sourceName>
      <sourceType logicalType="SINGLE_LINE_STRING" mandatory="false">
      </sourceType>
    </sourceReferenceSchemaFragment>

  </sourceReferenceSchemaFragments>
</tns:sourceReferenceSchema>
```

Securing i2 Analyze

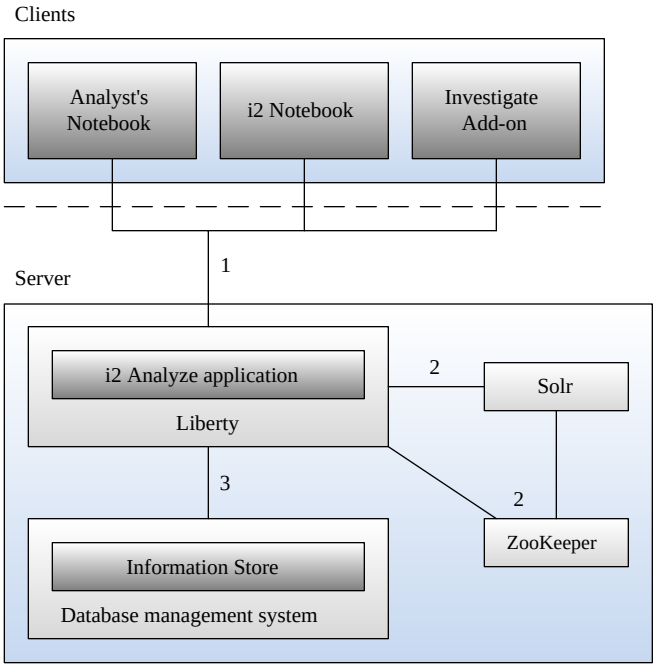
Depending on the requirements of the deployment, you can configure your deployment to use additional security mechanisms.

Transport Layer Security connections with i2 Analyze

Transport Layer Security (TLS) technology can be used to establish an encrypted connection between a client and server. You can use TLS to ensure that communication between i2 Analyze components is encrypted. By default, i2 Analyze supports version 1.2 of the TLS protocol, but you can configure it to support other versions too.

The physical architecture and the network topology of an i2 Analyze deployment determine how appropriate it is to use TLS to secure its connections. For example, if two parts of the deployment are on a single server that you physically control, then the need for TLS to secure the connection between them might be reduced. If your deployment contains a firewall, then you might need to weigh the benefits of inspecting traffic against the benefits of encrypting it.

You can configure TLS for the following connections in i2 Analyze:



The numbered connections in the diagram are as follows:

1. The connection between the clients and Liberty.
2. The connections between Liberty, ZooKeeper, and Solr.
3. The connection between Liberty and the database management system.

To secure connections 2 and 3, you must first secure connection 1.

Note: To enable access to i2 Analyze through the i2 Notebook web client, you *must* use TLS to secure the connection between the browser and Liberty. i2 Notebook requires the HTTPS protocol.

For information about securing the connection between Liberty and any i2 Connectors, see client authenticated Transport Layer Security with the i2 Connect gateway.

Important: At this release, if you configure Solr to use TLS after it is deployed without it, you must re-create your Solr collections. Aim to configure Solr to use TLS before you ingest a large amount of data into your system.

Attention: i2 takes reasonable steps to verify the suitability of i2 Analyze for internet deployment. However, it does not address lower-level issues such as guarding networks against penetration, securing accounts, protecting against brute force attacks, configuring firewalls to avoid DoS or DDoS attacks, and the like. For your deployment of i2 Analyze, follow industry-standard practices and recommendations for protection of your systems. i2 accepts no liability for the consequences of such attacks on your systems. This information is not intended to provide instructions for managing key databases or certificates.

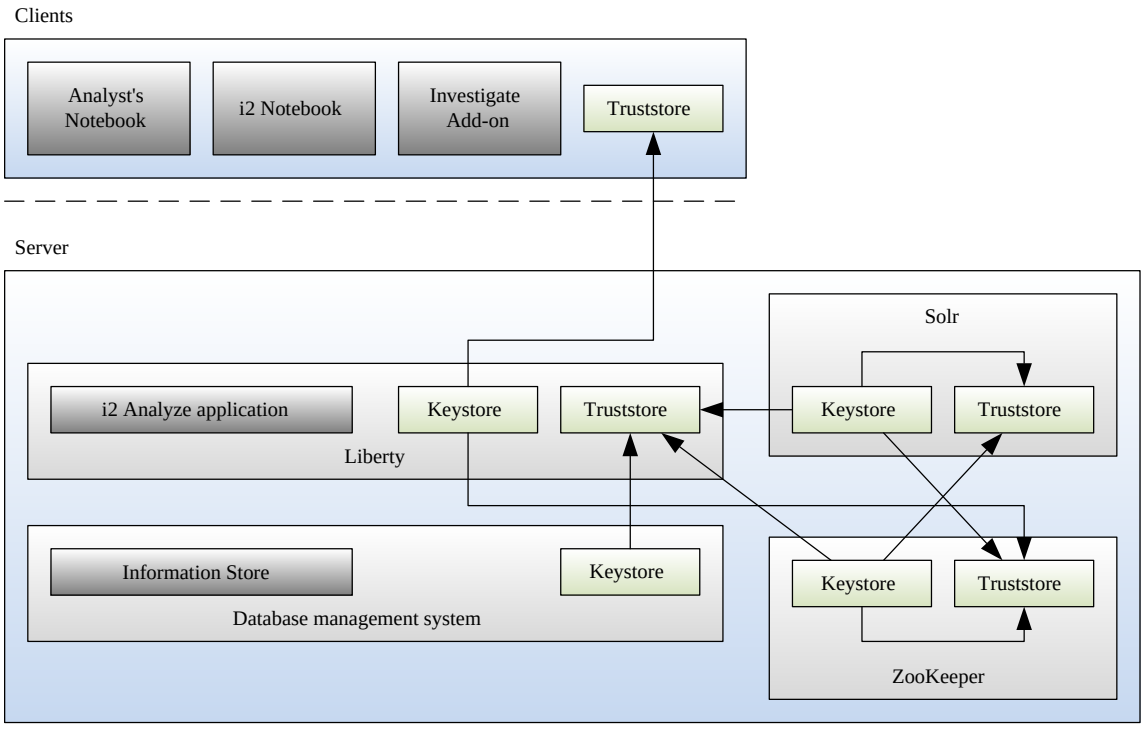
Keystores, truststores, and certificates for i2 Analyze

TLS communication relies on encryption, keys, and certificates to initiate a secure connection. The certificates are stored in keystore files on the client and the servers.

Certificates are exchanged to establish trust during the handshake process that initiates a secure connection. When a certificate is granted through a certificate authority, that certificate can be trusted by the clients or applications that trust certificates that are signed by that authority. A public key certificate that authenticates a server is stored in a keystore file on the server. Trusted certificate authority certificates are stored in the client's truststore file.

The environment where you are deploying i2 Analyze might already have a well-defined certificate process that you can use to obtain certificates and populate the required keystores and truststores.

The following diagram shows where the keystores and truststores are used in the components of i2 Analyze, and the connections between them.



The environment in which you are deploying i2 Analyze might already contain files that are candidates for use as keystores or truststores. If not, you must create the required files.

The files that are required are summarized by component in the following list. In the list, all certificates are described as being signed by the same certificate authority (CA).

Open Liberty

Each Liberty server requires:

- Keystore contents:
  - Liberty server private key
  - The personal certificate issued for the Liberty server by the certificate authority
- Truststore contents:
  - The CA certificate for the signing certificate authority

Solr

Each Solr server requires:

- Keystore contents:
  - Solr server private key
  - The personal certificate issued for the Solr server by the certificate authority

- Truststore contents:
  - The CA certificate for the signing certificate authority

ZooKeeper

Each ZooKeeper server requires:

- Keystore contents:
  - ZooKeeper server private key
  - The personal certificate issues for the ZooKeeper server by the certificate authority
- Truststore contents:
  - The CA certificate for the signing authority

Database management system

The type of keystore depends on the type of database management system. For more information about TLS in your database management system, see Configuring TLS for a Db2 instance, Configuring TLS for Microsoft SQL Server or Configuring TLS for PostgreSQL.

- Keystore contents:
  - The personal certificate issues for the database management system by the certificate authority

Configuring Liberty for TLS

To secure the connections between Open Liberty and the components of i2 Analyze, you must configure Liberty for TLS. This topic describes how to update the i2 Analyze configuration with the location of a keystore and truststore to use, and how to provide the passwords for accessing the certificates that the stores contain.

Before you begin

Before you can configure i2 Analyze, you must have a keystore and truststore for Liberty that contain the required certificates. For more information about the required certificates, see Keystores, truststores, and certificates for i2 Analyze.

About this task

Configuring Liberty to use TLS involves modifying the i2 Analyze `topology.xml` file to specify that a secure connection must be used with the application server. You must also update the `credentials.properties` file to specify the password for the Liberty keystore and truststore files.

By default, secure connections between i2 Analyze and its clients use TLS v1.2. To change this setting without affecting the other connections, you can edit the `environment-advanced.properties` file.

When configuration is complete, it is only possible to connect to Liberty through the HTTPS protocol, on the secure port that is defined in the port definition properties file. The non-secure port cannot be used.

Procedure

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   a. In the `<application>` element for the application server to secure, add the `secure-connection` attribute with the value of `true`.

      For example:

```
<application name="opal-server" host-name="hostname" secure-
connection="true">
```

Note: The `host-name` attribute value must match the common name that is associated with the certificate for the application server.

b. Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.

For your keystore, specify `type` as `key-store`, and `file` as the full path to your keystore. For your truststore, specify `type` as `trust-store`, and `file` as the full path to your truststore.

For example:

```
<application name="opal-server" host-name="hostname" secure-
connection="true">
...
  <key-stores>
    <key-store type="key-store"
        file="C:/i2/i2analyze/i2-liberty-keystore.p12"/>
    <key-store type="trust-store"
        file="C:/i2/i2analyze/i2-liberty-truststore.p12"/>
  </key-stores>
...
</application>
```

2. Specify the keystore passwords in the credentials file.

a. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.

b. Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

```
ssl.keystore.password=password
ssl.truststore.password=password
```

3. (Optional) Change or add to the list of supported TLS protocols for connections from clients to Liberty.

a. In a text editor, open the `toolkit\configuration\environment\environment-advanced.properties` file.

b. Follow the guidance in that file to set the value of the `wlp.sslProtocol` property.

For example, to enable TLS v1.3 as well as TLS v1.2, set the value to `TLSv1.2,TLSv1.3`.

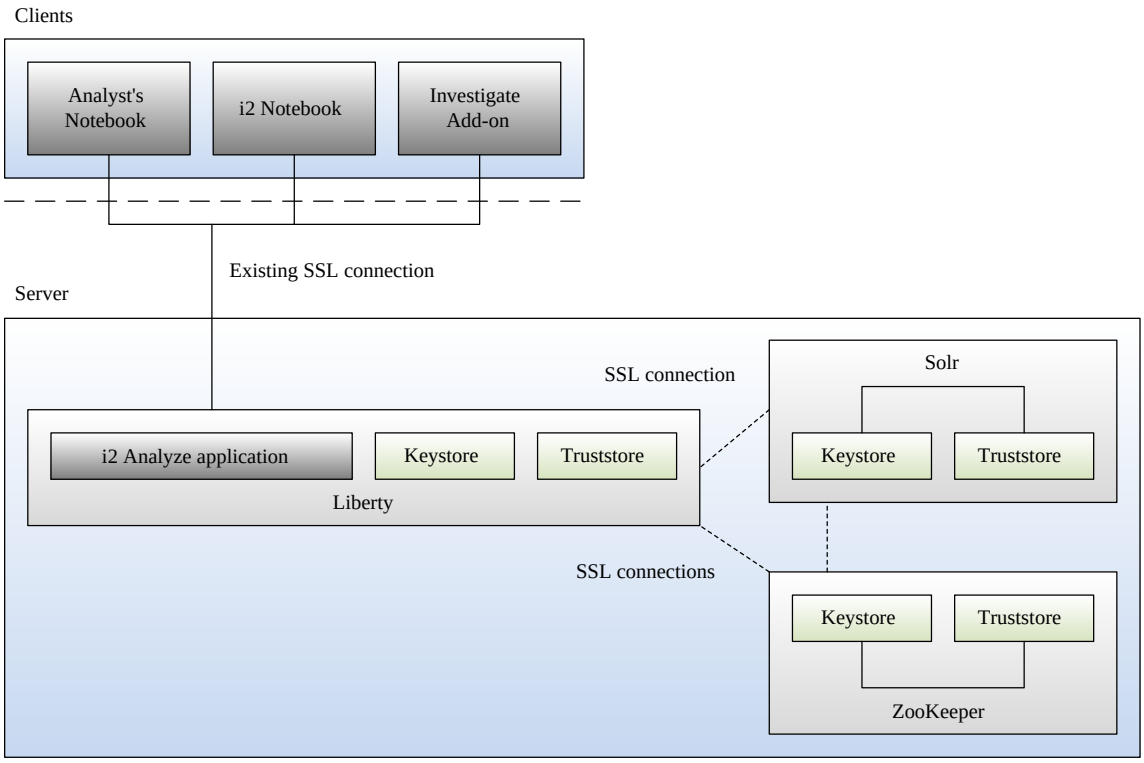
4. Update the application with your configuration changes. For more information, see Redeploying Liberty.

Securing the connection between Liberty, Solr, and ZooKeeper

You can secure the connection between Liberty, Solr, and ZooKeeper by using TLS.

Before you can secure the connection, Liberty must already be configured for TLS. For more information, see Configuring Liberty for TLS.

Setting up a TLS connection to Solr and ZooKeeper is a three-part process. First, configure Solr to use TLS, then configure ZooKeeper to use TLS, and finally update the i2 Analyze configuration with the location of the Solr and ZooKeeper keystores and passwords.



Configuring the TLS connections between Liberty and Solr and ZooKeeper

To secure the connection between the i2 Analyze application server, Solr, and ZooKeeper, you must change the configuration of all three components. The i2 Analyze configuration must define the keystore and truststore for Solr and ZooKeeper.

Before you begin

- Ensure that you configured Liberty for TLS. For more information, see Configuring Liberty for TLS.
- You must have the appropriate keystore set up for your Solr and ZooKeeper deployments.
  - For more information, see Keystores, truststores, and certificates for i2 Analyze.

About this task

Liberty uses a Java truststore to verify the certificate from the Solr and ZooKeeper servers.

Procedure

1. Stop Liberty by running the following command on the Liberty server:

   ```
   setup -t stopLiberty
   ```

Ensure that the Liberty instance is stopped, otherwise you encounter an error if you try to run the command when you complete the configuration changes.

2. Stop ZooKeeper by running the following command on every server where ZooKeeper is running:

   ```
   setup -t stopZkHosts --hostname zookeeper.host-name
   ```

   Here, `zookeeper.host-name` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the topology.xml file.

3. Ensure the Liberty, Solr, and ZooKeeper keystores and truststores contains the certificate required to authenticate the other components.

4. Modify the `topology.xml` file to specify TLS for its Solr connection.

   a. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

   b. In the `<solr-cluster>` element for the Solr cluster that you want to connect to with TLS, add the `secure-connection` attribute with the value of true.

      For example:

      ```
      <solr-cluster id="is_cluster" zookeeper-id="zoo" secure-
      connection="true">
      ```

   c. Add the `key-store` and `trust-store` attributes to either the `<solr-cluster>` or the `<solr-node>` element.

      Add the attribute values as defined:

      key-store

      The path to the Solr keystore.

      trust-store

      The path to the Solr truststore.

      For example, add the attribute as highlighted in the `<solr-clusters>` element:

      ```
      <solr-cluster id="is_cluster" zookeeper-id="zoo" secure-
      connection="true"
        key-store="C:\i2\i2analyze\i2-solr-keystore.p12" trust-store="C:
      \i2\i2analyze\i2-solr-truststore.p12">
      ```

      Or add the attribute as highlighted in the `<solr-node>` element:

      ```
      <solr-node memory="2g" data-dir="C:\i2\i2analyze\data\solr" host-
      name="hostname" id="node1" port-number="8983"
        key-store="C:\i2\i2analyze\i2-solr-keystore.p12" trust-store="C:
      \i2\i2analyze\i2-solr-truststore.p12">
      ```

      Note: The `host-name` attribute value must match the common name that is associated with the certificate for Solr.

5. Modify the `topology.xml` file to specify TLS for its ZooKeeper connection.

   a. In the `<zookeeper>` element for the ZooKeeper host that you want to connect to with TLS, add the `secure-connection` attribute with the value of `true`.

      For example:

      ```
      <zookeeper id="zoo" secure-connection="true">
      ```

   b. Add the `key-store` and `trust-store` attributes to either the `<zookeeper>` or to the `<zkhost>` element.

Add the attribute values as defined:

key-store

The path to the ZooKeeper keystore.

trust-store

The path to the ZooKeeper truststore.

- For example, add the attributes in the `<zookeeper>` element:

```
<zookeeper id="zoo" secure-connection="true" key-store="C:
\i2\i2analyze\i2-zookeeper-keystore.p12" trust-store="C:\i2\i2analyze
\i2-zookeeper-truststore.p12">
```

- For example, add the attributes in the `<zkhost>` element:

```
<zkhost quorum-port-number="10483" leader-port-number="10983"
 data-dir="C:\i2\i2analyze\data\zookeeper" host-name="hostname"
 id="1" port-number=" 9983" key-store="C:\i2\i2analyze\i2-
zookeeper-keystore.p12" trust-store="C:\i2\i2analyze\i2-zookeeper-
truststore.p12">
```

Note: The `host-name` attribute value must match the common name that is associated with the certificate for ZooKeeper.

6. Specify the truststore and keystore passwords in the credentials file.
   a. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter the passwords for the Solr keystore and truststore that you specified in the topology file.

```
solr.truststore.password=password
solr.keystore.password=password
```

   c. Enter the passwords for the ZooKeeper keystore and truststore that you specified in the topology file.

```
zookeeper.truststore.password=password
zookeeper.keystore.password=password
```

7. Copy the `toolkit\configuration` from the Liberty server, to the toolkit directory of the deployment toolkit on each server in your environment.
8. Update the application with your configuration changes. Run the following commands from the `toolkit\scripts` directory on the Liberty server.
   a. Redeploy Liberty to update the application:

```
setup -t deployLiberty
```

   b. Recreate the ZooKeeper host on each server where your ZooKeeper hosts are located:

```
setup -t createZkHosts --hostname zookeeper.host-name
```

   Where `zookeeper.host-name` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the topology.xml file.

   c. Start ZooKeeper.

   To start ZooKeeper, run the following command on every server where your ZooKeeper hosts are located:

```
setup -t startZkHosts --hostname zookeeper.host-name
```

Where `zookeeper.host-name` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the topology.xml file.

d. Upload the new Solr configuration to ZooKeeper:

```
setup -t createAndUploadSolrConfig --hostname liberty.hostname
```

Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the topology.xml file.

e. Restart the Solr nodes.

To restart the Solr nodes, run the following command on every server where Solr is running:

```
setup -t restartSolrNodes --hostname solr.host-name
```

Where `solr.host-name` is the host name of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

f. Start Liberty.

To start Liberty, run the following command on each Liberty server:

```
setup -t startLiberty
```

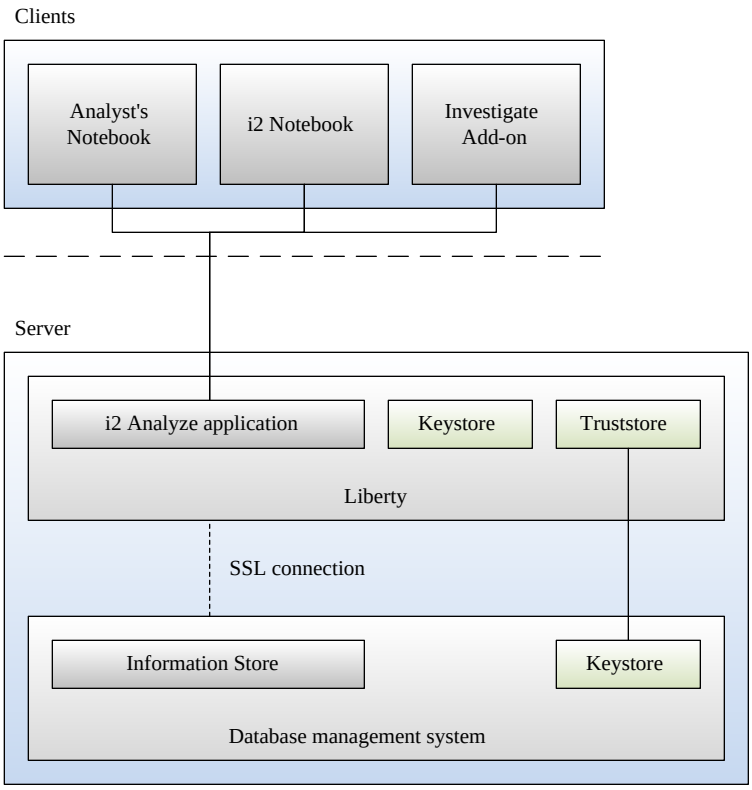
Securing the connection between Liberty and the database

In an i2 Analyze deployment, Liberty connects to the database management system. You can secure the connection between Liberty and the database management system by using TLS.

Before you can secure the connection, Liberty must already be configured for TLS. For more information, see Configuring Liberty for TLS.

Setting up an TLS connection to the database is a two-part process. First, you must configure your database instance to use TLS. Then, regardless of which database management system you use, you must update the i2 Analyze configuration with the location of the truststore and the truststore password.

This diagram shows the connection that you secure by configuring TLS. It also shows the keystore and truststore that are required.



Note: Instead of employing a truststore, deployments that use PostgreSQL refer directly to a certificate file. If you're using the PostgreSQL RDBMS, read Configuring SSL for a PostgreSQL instance rather than any of the other topics below.

Configuring TLS for a Db2 instance

To secure the connection between the i2® Analyze application server and the database instance, you must change the configuration of both. The Db2® server stores its associated certificates in a key database, so you must create and populate a key database for the Db2 server to use.

Before you begin

Ensure that you configured Liberty for TLS. For more information, see Configuring Liberty for TLS.

About this task

In i2 Analyze, TLS connections that involve the Db2 server require a key database that contains a signed certificate. In a production deployment, after you create the key database, you must populate it with a certificate that is signed by a trusted certificate authority.

Procedure

1. Follow the steps that are provided in the Db2 documentation to configure TLS for your Db2 instance. For more information, see Configuring Secure Sockets Layer (TLS) support in a Db2 instance.

Note: If you are using a remote Db2 database, then you must configure the Db2 client on the Liberty server to communicate by using TLS.

Configuring TLS for Microsoft SQL Server

To secure the connection between the i2® Analyze application server and the database instance, you must change the configuration of both. Microsoft™ SQL Server stores its associated certificates and you must create or obtain certificates for the Microsoft SQL Server to use.

Before you begin

Ensure that you configured Liberty for TLS. For more information, see Configuring Liberty for TLS.

About this task

In i2 Analyze, TLS connections that involve SQL Server require i2 Analyze to trust the certificate that it receives from SQL Server. SQL Server stores certificates in the operating system's certificate stores. In a production deployment, you must use a certificate that is signed by a trusted certificate authority.

Procedure

1. Follow the steps that are provided in the SQL Server documentation to configure TLS for your SQL Server. For more information, on Windows™ see Configure SQL Server Database Engine for encrypting connections or Linux® see Encrypting Connections to SQL Server on Linux.

Configuring i2 Analyze to connect to a database instance using TLS

To connect to a database instance by using TLS, i2 Analyze must be able to authenticate the certificate that it receives from the database server.

Before you begin

- Ensure that you configured Liberty for TLS. For more information, see Configuring Liberty for TLS.
- Your database management system must be configured for TLS. For more information, see:
  - Configuring TLS for a Db2 instance
  - Configuring TLS for Microsoft SQL Server

Important: If your deployment of i2 Analyze uses PostgreSQL rather than IBM Db2 or Microsoft SQL Server, read Configuring SSL for a PostgreSQL instance instead of this topic.

About this task

i2 Analyze uses a Java™ truststore to verify the certificate from the database server, and so you must create a truststore on your i2 Analyze server that contains the trusted certificates for your database. You can use the same truststore that is used for Liberty. For more information, see Creating the Liberty keystore and certificate.

For Liberty to communicate with the secured database, in the topology database element you must specify the secure connection attribute to be true and the name of the truststore that contains the database certificate. Also, specify the correct port number, which corresponds to the TLS port for the

database. In the `credentials.properties` file, the correct password for the specified truststore must be added.

Procedure

1. Modify the i2 Analyze topology to use TLS for its database connection.
   a. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   b. In the `<database>` element for the database that you want to connect to with TLS, add the `secure-connection="true"` attribute.
   c. In the same `<database>` element, add the `trust-store` attribute with the location of the truststore.

      If you are using Db2:

      ```
      <database database-type="InfoStore" dialect="db2"
                instance-name="DB2" database-name="ISTORE" xa="false"
                id="infostore" host-name="hostname" port-number="50001"
                secure-connection="true"
                trust-store="C:/i2/i2analyze/i2-liberty-truststore.p12"/>
      ```

      If you are using SQL Server:

      ```
      <database database-type="InfoStore" dialect="sqlserver"
                database-name="ISTORE" xa="false" id="infostore"
                host-name="hostname" port-number="1433"
                secure-connection="true"
                trust-store="C:/i2/i2analyze/i2-liberty-truststore.p12"/>
      ```

   d. In same element, ensure that the following attribute values are correct:
      - The `host-name` attribute value must match the common name that is associated with the certificate for the database.
      - The `port` attribute value must match the value of the port number when you configured the database management system for TLS.
2. Specify the truststore password in the credentials file:
   a. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter the password for the truststore in the `db.infostore.truststore.password` credential.

Update the application with your configuration changes. For more information, see Redeploying Liberty.

Configuring SSL for a PostgreSQL instance

To secure the connection between the i2® Analyze application server and the database instance, you must enable SSL on the PostgreSQL database server, then configure and deploy the toolkit.

Enable support for remote connections

By default, PostgreSQL deployments support connections only from localhost. For use in a production environment, you should enable remote connections before you enable SSL.

You can find the files that you need to modify, `pg_hba.conf` and `postgresql.conf`, in the `data` folder on the PostgreSQL database server.

To allow connection from any host to all PostgreSQL databases, using any PostgreSQL user and password, add these lines to the end of the `pg_hba.conf` file:

```
host    all     all     0.0.0.0/0       scram-sha-256
host    all     all     ::/0            scram-sha-256
```

If you need to set restrictions, see the instructions at the top of the `pg_hba.conf` file and modify the file as required.

In the `postgresql.conf` file, set the `listen_addresses` to `'*'` to allow connection from any host. If you need to set restrictions, see the comments for `listen_addresses` in the `postgresql.conf` file and modify the file as required. After you modify this file, restart Postgres to apply the changes.

Note: When you allow connections from *any* host, this includes connections from localhost.

Open the PostgreSQL port in the firewall. Before you enable SSL, check the changes are working by redeploying i2 Analyze. For more information, see Redeploying Liberty.

Enable SSL on the PostgreSQL database server

Follow the steps in the PostgreSQL documentation to configure SSL for your PostgreSQL instance. For more information, see Secure TCP/IP Connections with SSL.

Configure the toolkit and deploy

After you enable PostgreSQL to use SSL, you can configure i2 Analyze to connect securely. You'll need the Postgres server's public certificate file that was used when you enabled SSL on the database server.

1. Place the certificate file in a suitable folder on the Liberty server.
2. Edit the `topology.xml` file and add the following settings to the Information Store-related database node, where `<public certificate file-path>` is the absolute file path of the Postgres server's public certificate file:

   ```
   secure-connection="true" trust-store="<public certificate file-path>"
   ```

   For example:

   ```
   <databases>
     <database database-type="InfoStore" dialect="postgres"
               database-name="ISTORE" instance-name="" xa="false"
               id="infostore" host-name="hostname" port-number="5432"
               secure-connection="true" trust-store="C:/i2/server.crt"/>
   </databases>
   ```

   Note: For PostgreSQL, the `trust-store` value means the location of the server's public certificate itself, not the location of a password-protected trust-store file that contains the certificate.
3. To deploy the SSL configuration changes, run:

   ```
   setup -t stopLiberty
   setup -t deployLiberty
   setup -t startLiberty
   ```

When the Liberty server is started and it connects to the Information Store database, it verifies the certificate obtained from the PostgreSQL database server, using the local CA certificate file. If verification fails, it fails to connect to the database. For more information, see SSL Support.

If you are planning to use the ETL toolkit, you must re-create the ETL toolkit after you configure SSL. For more information, see Deploying the ETL toolkit.

Installing certificates on client workstations

To enable TLS connections to i2® Analyze, the certificate the client receives from Liberty must be trusted on each client workstation. To trust the certificate, the client workstation must have the certificate authority's certificate installed.

The following steps describe how to install a certificate on a Windows client. To install the certificate on Linux workstations, see the documentation for your operating system.

1. Copy the certificate to any folder on the client workstation.
2. To install the certificate, complete the following steps:
   a. Double-click the certificate file.
   b. Click **Install Certificate**, and then click **Next**.
   c. Click **Place all certificates in the following store**.
   d. Click **Browse**, select **Trusted Root Certification Authorities**, and click **OK**.
   e. Click **Next**, and then click **Finish**.

Creating keystores, truststores, and certificates for development

You can use the Java keytool to create keystores and certificates for development purposes. For more information about Java keytool, see keytool - Key and Certificate Management Tool.

About this task

The certificates that the Java keytool generates are self-signed and they should not be used in production deployments. You can use the following instructions to create the keystores, truststores, and certificates for Liberty, Solr, and ZooKeeper in a development environment. When you are deploying into production, you must use certificates that are provided by a trusted certificate authority.

For more information about the location of the required stores and certificates, see Keystores, truststores, and certificates for i2 Analyze.

Procedure

1. Create a keystore and self-signed certificate.
   a. Open a command prompt and navigate to the Java bin directory on the server.
   For example, on the Liberty server: `i2analyze\deploy\java\bin`
   b. Create a keystore and certificate.

   For example, run the following command:

   ```
   keytool -genkeypair -alias "libertyKey" -keystore "C:\i2\i2analyze
   \i2-liberty-keystore.p12" -dname "CN=hostname" -keyalg RSA -storepass
    "password" -ext san=dns:hostname
   ```

   Important: Ensure that you provide values as follows:

   - Enter a unique alias.
   - Set the value of `CN` and `san=dns` to the hostname of the server.
   - Assign a secure password.

   c. Export the certificate from the keystore.

   For example, run the following command:

```
keytool -exportcert -alias "libertyKey" -keystore "C:\i2\i2analyze\i2-
liberty-keystore.p12" -file "C:\i2\i2analyze\i2-liberty-certificate.cer"
 -storepass "password"
```

When you are using self-signed certificates, you must add the certificates that you exported from your keystores to the required truststores.

1. Create a truststore and import the specified certificate into the truststore.

   For example, run the following command:

   ```
   keytool -importcert -alias "solrKey" -file "C:\i2\i2analyze\i2-liberty-
   certificate.cer" -keystore "C:\i2\i2analyze\i2-liberty-truststore.p12" -
   storepass "password"
   ```

   Enter yes in response to the query, `Trust this certificate?`

   Important: Ensure that you enter values as follows:

   - Enter a unique alias.
   - Assign a secure password.

If the truststore already exists, the command adds the certificate to the existing truststore.

Testing TLS in an i2 Analyze deployment

To ensure that a deployment allows only connections that use TLS, you can access it from a client workstation.

- The certificate that the client receives from Liberty server must be trusted on the client workstation. For more information, see Installing the certificate on client workstations.
- Ensure that i2 Analyze is started.
- Make a note of the URI for connections. When you start i2 Analyze, the URI that can be used to connect is displayed in the console. For example:

  ```
  Web application available (default_host): https://host_name:9445/opal
  ```

  Note: You cannot connect by using the HTTP protocol `http://host_name/opal`.

To use Analyst's Notebook to connect, you must:

- Install i2 Analyze support in Analyst's Notebook. For more information, see Installing i2 Analyst's Notebook.
- Connect to the deployment. For more information, see Working with i2 Analyze.

To use a web client to connect, you must:

- Ensure that you have the correct license agreements in place to use the i2 Investigate or the i2 Notebook web client.
- Open a web browser, and navigate to `https://host_name/opal`. The web client displays a login dialog.
- Enter the name and password of a user who is registered in the application server.

  If that user has permission to use the i2 Notebook web client, you see that application's user interface. If they do not have permission, you see the i2 Investigate web client user interface instead.

Resources for system protection

In order to protect your system from external forces, you must implement system controls that prevent or mitigate the effect of attacks. Although i2 Group does not manage login configuration, and the responsibility for protection of your network from external attack remains yours, the following communities provide a starting point for your investigation into preventative methods.

The Open Web Application Security Project

The Open Web Application Security Project Foundation is a not-for-profit organization that is dedicated to enabling organizations to conceive, develop, operate, and maintain applications that can be trusted.

In particular, see https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks and https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

SANS Institute

The System-Admin, Audit, Network, and Security Institute is the largest source for information security training and security certification in the world. It also develops, maintains, and makes available at no cost, the largest collection of research documents about various aspects of information security, and it operates the internet's early warning system - the Internet Storm Center.

In particular, see the Password Construction Guidelines in its Security Policy Templates.

Common Weakness Enumeration

CWE™ is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts.

In particular, see http://cwe.mitre.org/top25/index.html#CWE-307.

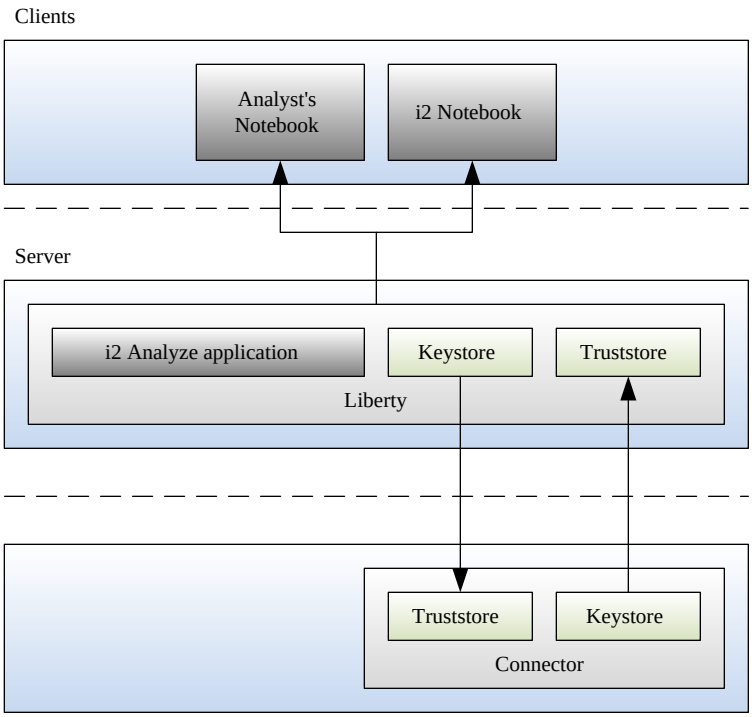
Client-authenticated Transport Layer Security with the i2 Connect gateway

To secure the connection between Liberty and any connectors for the i2 Connect gateway, you must configure Liberty and your connectors to use TLS. If you are using TLS, i2 Analyze enforces client-authenticated communication with a connector.

Before you begin

In a production deployment you should configure i2 Analyze to connect to your connector using client-authenticated TLS communication. To do so, your connector and i2 Analyze must trust the certificates that they receive during the TLS handshake process. In a production environment, the certificates must be signed by a trusted certificate authority. For more information about client authenticated SSL, see Client-authenticated TLS handshake.

The following diagram shows the keystores and truststores that are required for Liberty and the connector.



The Liberty server requires a keystore file and a truststore file. Your connector can use any implementation for its keystore and truststore. The certificates in each truststore must trust the certificates received from the corresponding keystore.

The certificates that are required are as follows, where certificate authority (CA) X issues the certificates to the connector (the server) and Liberty (the client):

Connector

Each connector requires:

- Keystore contents:
  - The personal certificate issued to the connector by CA X
  - The connector's private key
- Truststore contents:
  - The CA certificate for CA X

Open Liberty

Each Liberty server requires:

- Keystore contents:
  - The personal certificate issued to Liberty by CA X
  - Liberty's private key

- Truststore contents:
  - The CA certificate for CA X

About this task

After you have created and populated the keystores and truststores for the connector and Liberty, you must configure Liberty to use TLS to communicate with any connectors.

The following steps explain the process of updating the i2 Analyze configuration with the location of a keystore and truststore to use, and the passwords that are used to access the certificates that are contained within them.

To configure the example connector to use client-authenticated TLS, and for examples of how to create keystores, truststores, and certificates for Liberty, follow the instructions in Securing the example connector.

Procedure

1. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.
   a. Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.

      For your keystore, specify the `type` as `key-store`, and `file` as the full path to your keystore. For your truststore, specify the `type` as `trust-store`, and `file` as the full path to your truststore.

      For example, add the attribute as highlighted in the following code:

      ```
      <application http-server-host="true"
          name="opal-server" host-name="<hostname>">
        ...
        <key-stores>
          <key-store type="key-store"
              file="C:/i2/i2analyze/i2-liberty-keystore.jks"/>
          <key-store type="trust-store"
              file="C:/i2/i2analyze/i2-liberty-truststore.jks"/>
        </key-stores>
        ...
      </application>
      ```

   b. Update the `base-url` attribute of any connectors using TLS to use the HTTPS protocol.

      For example:

      ```
      <connectors>
        <connector id="example-connector" name="Example"
            base-url="https://localhost:3700/"/>
        </connector>
      ```

      Note: Ensure that the hostname that is used in the base URL matches the common name on the certificate of the connector.
2. Specify the keystore and truststore passwords in the credentials file.
   a. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
   b. Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

      ```
      ssl.keystore.password=<password>
      ssl.truststore.password=<password>
      ```

3. Redeploy i2 Analyze to update the application with your changes.
   a. In a command prompt, navigate to the `toolkit\scripts` directory.
   b. Stop Liberty:
      ```
      setup -t stopLiberty
      ```
   c. Update the i2 Analyze application:
      ```
      setup -t deployLiberty
      ```
   d. Start Liberty:
      ```
      setup -t startLiberty
      ```

What to do next

You can create your own connectors to use with the deployment of i2 Analyze, when you create your own connector you can implement security that conforms to the security required by the i2 Connect gateway. For more information about creating your own connectors, see i2 Analyze and i2 Connect.

When you use a connector configured for TLS communication, you should not see any warnings displayed in Analyst's Notebook.

Securing the example connector

In a production deployment, you must configure your connectors with client-authenticated TLS communication. You can configure the example connector to communicate with i2 Analyze by using client-authenticated TLS communication.

Before you begin

You must configure your connectors to use client-authenticated TLS communication with i2 Analyze. For more information about configuring security between connectors and i2 Analyze, see Client-authenticated Transport Layer Security with the i2 Connect gateway.

About this task

Configure secure communication between Liberty and the example connector that i2 provides as an npm package. The following steps demonstrate how to create the keystores, truststores, and certificates, then configure i2 Analyze and the example connector to use them.

When you create your own connector, you can implement the security implementation in the same way as used in the example connector, or a different method depending on your requirements.

In a production deployment, you must use a certificate that has been signed by a trusted certificate authority, which the connector can verify. The following steps use a self-signed certificate to demonstrate the functionality.

Procedure

1. Use the Java keytool utility to create a keystore for Liberty that contains a signed certificate for authenticating with the connector. (For more information about the Java keytool utility, see keytool - Key and Certificate Management Tool).
   a. Open a command prompt and navigate to the `i2analyze\deploy\java\bin` directory.
   b. If you haven't already done so, create the keystore and certificate by running the following command:

```
keytool -genkeypair -alias "libertyKey"
-keystore "C:\i2\i2analyze\i2-liberty-keystore.jks"
-dname "CN=<hostname>" -keyalg RSA -storepass "libertyKeyStorePassword"
```

Important: Ensure that you enter values as follows:

- Set the location of the keystore to the directory that contains the toolkit.
- Set the value of `CN` to the hostname of the server that hosts Liberty, as defined in the `topology.xml` file.

The password that is specified is used to access the keystore.

c. The example connector implementation requires the Liberty certificate in a base64 encoded format. Export the certificate in base64 encoding by running the following command:

```
keytool -exportcert -alias "libertyKey"
-keystore "C:\i2\i2analyze\i2-liberty-keystore.jks"
-file "C:\i2\i2analyze\i2-liberty-certificate.pem"
-rfc -storepass "libertyKeyStorePassword"</codeblock>
```

2. Use the Java keytool utility to create a keystore for the connector that contains a signed certificate for authenticating with Liberty.

a. You need a place to store the connector files. Create a configuration directory for the example connector in the toolkit:

```
mkdir "C:\i2\i2analyze\toolkit\configuration\example-connector"
```

b. Create the keystore and certificate in PKCS12 format (the default). To create the keystore, run the following command:

```
keytool -genkeypair -alias "connectorKey" -keystore "C:\i2\i2analyze
\toolkit\configuration\example-connector\example-connector-
keystore.p12" -dname "CN=<hostname>" -keyalg RSA -storepass
 "connectorKeyStorePassword"
```

The password that is specified is used to access the keystore.

Important: Set the value of `CN` to the hostname of the server that hosts the connector, as defined in the `topology.xml` file. By default in the `daod-opal` example configuration, the value is `localhost`.

3. The example connector requires the private key in the PEM format, and the certificate for the connector.

a. Convert the private key from the PKCS12 format to the PEM format. You can use OpenSSL to do this. For more information about OpenSSL, see https://www.openssl.org/source. On Windows you can use easy-rsa to run OpenSSL commands, see https://github.com/OpenVPN/easy-rsa. If you are using OpenSSL, you can run the following command:

```
openssl pkcs12 -in C:/i2/i2analyze/toolkit/configuration/example-
connector/example-connector-keystore.p12
-out C:/i2/i2analyze/toolkit/configuration/example-connector/example-
connector-key.pem
-nocerts -nodes
```

When you are prompted for a password, provide the password that you specified when you created the `example-connector-keystore.p12` PKCS12 file. In this example, enter `connectorKeyPassword`.

b. Extract the certificate for the connector by running the following command:

```
keytool -exportcert -alias "connectorKey" -keystore "C:\i2\i2analyze
\toolkit\configuration\example-connector\example-connector-keystore.p12"
```

```
 -file "C:\i2\i2analyze\toolkit\configuration\example-connector\example-
connector-certificate.pem" -rfc -storepass "connectorKeyStorePassword"
```

Note: The `example-connector-keystore.p12` file is not used by the connector and can be removed now that you have extracted the key and certificate.

4. Use the Java keytool utility to create a truststore for Liberty that contains the certificate that is used to trust the connector:

```
keytool -importcert -alias "exampleconnector"
-keystore "C:\i2\i2analyze\i2-liberty-truststore.jks"
-file "C:\i2\i2analyze\toolkit\configuration\example-connector\example-
connector-certificate.pem"
-storepass "libertyTrustStorePassword"
```

In response to the prompt, enter `yes` to trust the certificate.

5. The example connector implementation requires the certificate that is used to trust Liberty to be present in the same directory. Copy the `C:\i2\i2analyze\i2-liberty-certificate.pem` file to the `C:\i2\i2analyze\toolkit\configuration\example-connector` directory.

The certificates are now in place to enable client-authenticated SSL between Liberty and the connector.

6. Next, configure Liberty to use the keystore and truststore that you created. In an XML editor, open the `toolkit\configuration\environment\topology.xml` file.

   a. Add the `<key-stores>` element as a child of the `<application>` element. Then, add child `<key-store>` elements.
   For your keystore, specify the `type` as `key-store`, and `file` as the full path to your keystore.
   For your truststore, specify the `type` as `trust-store`, and `file` as the full path to your truststore.
   For example, add the element in the following code:

```
<application http-server-host="true"
  name="opal-server" host-name="hostname">
...
  <key-stores>
      <key-store type="key-store"
        file="C:/i2/i2analyze/i2-liberty-keystore.jks"/>
      <key-store type="trust-store"
        file="C:/i2/i2analyze/i2-liberty-truststore.jks"/>
  </key-stores>
...
</application>
```

   b. Update the `base-url` attribute of any connectors using TLS to use the HTTPS protocol. For example:

```
<connectors>
  <connector id="example-connector" name="Example"
    base-url="https://localhost:3700/" />
</connector>
```

   Note: Ensure that the hostname that is used in the base URL matches the common name on the certificate of the connector.

7. Specify the keystore and truststore passwords in the credentials file.

   a. In a text editor, open the `toolkit\configuration\environment\credentials.properties` file.

   b. Enter the password for the keystore and truststore that you specified in the `topology.xml` file.

```
ssl.truststore.password=libertyTrustStorePassword
ssl.keystore.password=libertyKeyStorePassword
```

8. In a command prompt, navigate to the `toolkit\scripts` directory.
9. Update the i2 Analyze application:

   ```
   setup -t deployLiberty
   ```

10. Finally, configure, download, and start the example connector, and then start Liberty.
    a. In a *new* command prompt, navigate to the `toolkit\configuration\example-connector` directory.
    b. Create a file named `.env` in the `toolkit\configuration\example-connector` directory, with the following contents:

       ```
       SSL_ENABLED=true
       SSL_SERVER_PORT=3700
       SSL_PRIVATE_KEY_FILE="example-connector-key.pem"
       SSL_PASSPHRASE="connectorKeyPassword"
       SSL_CERTIFICATE_FILE="example-connector-certificate.pem"
       SSL_CA_CERTIFICATE_FILE="i2-liberty-certificate.pem"
       SSL_GATEWAY_CN="<hostname>"
       ```

       Note: You specified the value of `CN` in the certificate in Step 1.
    c. Install the dependencies and start the server that hosts the example connector.

       Note: The example connector uses port number 3700. Ensure that no other processes are using this port number before you start the connector.

       Execute the following command:

       ```
       npx @i2analyze/example-connector
       ```

       Note: You must be connected to the internet to download the package.
    d. Start Liberty:

       ```
       setup -t startLiberty
       ```

What to do next

Use Analyst's Notebook to connect to your deployment. For more information, see Working with i2 Analyze.

Now that the example connector is configured for TLS, no warnings are displayed in Analyst's Notebook.

Adding more data sources

You can extend your system to use different sources of data by adding the i2 Connect gateway, and configuring your deployment to identify the data source and searching to use. You can add the i2 Connect gateway to an existing deployment of i2 Analyze.

Before you begin

You must have a deployment of i2 Analyze. When you add the i2 Connect gateway to your deployment, the configuration for an example connector is included. To use the example connector, you must download and install Node.js to host the example connector. Download Node.js for your operating system from: https://nodejs.org/en/download/. You can install Node.js with the default settings.

To use a custom connector that is tailored to your deployment, you either need the details of a connector that is provided to you, or to create a connector. For more information about how to create a custom connector, see Connecting to external data sources.

About this task

The i2 Connect gateway enables analysts to search for and retrieve data from external data sources, and then analyze the results on a chart in Analyst's Notebook. To use the i2 Connect gateway, you must obtain or create a custom connector to the external data source that you want to search. The i2 Analyze toolkit contains an example configuration for the deployment, and i2 publishes a package that contains an example connector to www.npmjs.com.

Procedure

1. Run the `addI2Connect` task to add the i2 Connect gateway to your deployment. In a command prompt, navigate to the `toolkit\scripts` directory and run the following command:

   ```
   setup -t addI2Connect
   ```
2. Decide which connector you would like to add to your deployment:
   - If you want to use the example connector, complete the step to download and start the example connector in Deploying i2 Analyze with the Information Store and the i2 Connect gateway.
   - If you would like to add a custom connector, replace the following elements from the `configuration\environment\topology.xml` file with elements specific to the connector you want to use:
     - The `<connector>` element with the ID `example-connector`
     - The `<connector-id>` element with the value `example-connector`

     Note: For more information about making connector-specific changes to the topology file, see Adding a connector to the topology.
3. Redeploy i2 Analyze.

   ```
   setup -t deploy
   ```
4. Start i2 Analyze.

   ```
   setup -t start
   ```

What to do next

When you start i2 Analyze, the URI that users must specify in Analyst's Notebook is displayed in the console. For example, `This application is configured for access on http://<host_name>/opal`.

Install Analyst's Notebook and connect to your deployment. For more information, see Installing i2 Analyst's Notebook and Working with i2 Analyze.

Note: The example connector does not use client-authenticated SSL communication with i2 Analyze, so a warning is displayed in Analyst's Notebook. For more information about configuring client-authenticated SSL, see Client authenticated Secure Sockets Layer with the i2 Connect gateway.

Back up and restore

An effective backup strategy considers the key components in your deployment and assess the impact of creating replica versions versus the cost of replacing those components if the system fails. All production deployments of i2 Analyze should have a backup and restore strategy in place.

The following components of an i2 Analyze deployment should be backed up to provide a comprehensive backup strategy:

- The configuration
- The Solr search index and ZooKeeper configuration (If you have an Information Store or Chart Store)
- The database (If you have an Information Store or Chart Store)

When you create a backup strategy, consider the following items:

- The amount of time, if any, that is acceptable for a system to be offline to create backups.
- The amount of time that is acceptable for a system to be offline while recovery is in progress.
- The amount of time between backups, for each component.

In a deployment of i2 Analyze that contains a database, the database is the source of truth about the data contained in a deployment.

This means that the Solr search index must be backed up before the database is and that the database must be restored before the Solr search index. If any data in the Information Store was modified after the Solr search index was backed up, the changes are updated in the index after the restoring.

The i2 Analyze deployment toolkit contains toolkit tasks that you can use to back up and restore each component of i2 Analyze. You can choose to use the toolkit tasks to back up all components, or you can use the toolkit tasks to back up a subset and use the native back up and restore procedure provided by other components. For example, if your organization has existing processes for backing up and restoring databases but not a Solr search index, you can use the existing process to back up the database in a deployment and the provided toolkit task to back up the Solr search index.

Backing up a deployment

In a production deployment of i2 Analyze, use a tested backup and restore procedure to ensure that you can recover from failures. The various components of i2 Analyze must be backed up for complete backup coverage.

About this task

The following steps describe the process of backing up a deployment of i2 Analyze by using the provided toolkit tasks. You do not need to back up every component by using the toolkit tasks. For example, you can use the toolkit task to back up Solr and use existing processes to back up the database. Regardless of the mechanism that you use to back up the components, you must take the Solr backup before the database backup.

The configuration and Solr backups can be completed while the deployment is running. For more information, see:

- Back up and restore Solr
- Back up and restore the configuration

When you use the toolkit task to back up the database, you must stop the deployment first. To perform an online backup of the database, see the documentation for your database management system.

For more information about backing up the database using the toolkit, see: Back up and restore the database.

In a deployment that contains the i2 Connect gateway only, you only need to backup the i2 Analyze and Liberty configuration.

Run each toolkit command from a Liberty server in your deployment.

Procedure

1. In the `environment.properties` file, specify the locations where the toolkit creates the backup files.

   `backup.config.location.dir`
   The location where the i2 Analyze configuration backups are created and restored from. This location must exist on the Liberty server where you run the `backupConfiguration` command.

   `backup.solr.location.dir`
   The location where the Solr index and ZooKeeper configuration backups are created and restored from. This location must be accessible by every Solr node in your deployment.

   `backup.db.location.dir`
   The location where the database backups are created and restored from. The user that is specified in the `credentials.properties` file for your database management system must have write permissions to this location. This location must be accessible on the database server.

   For example:

   ```
   backup.config.location.dir=C:/backup/configuration
   backup.solr.location.dir=D:/network-drive/solr/backup
   backup.db.location.dir=D:/backup/database
   ```
2. To back up the i2 Analyze and Liberty configuration, run the following command from a Liberty server.
   In a deployment with high availability, run this command on each Liberty server.

   ```
   setup -t backupConfiguration
   ```
3. To back up the Solr search index, run the following command from a Liberty server:

   ```
   setup -t backupSolr
   ```
4. To back up the database in your deployment by using the toolkit task, you must stop the application first, and ensure that there are no other connections to the database, then run the backup command. In a deployment that contains multiple Liberty servers, you must run the `stopLiberty` command on every Liberty server.

   ```
   setup -t stopLiberty
   setup -t backupDatabases
   setup -t startLiberty
   ```

Restoring a deployment to a point in time

Use the backups of the components of i2 Analyze to restore the system to a previous point in time. In a production deployment of i2 Analyze, use a tested backup and restore procedure to ensure that you can recover from failures.

About this task

The following steps describe the process of restoring a deployment of i2 Analyze by using the provided toolkit tasks. You do not need to restore every component by using the toolkit tasks. For example, you

can use existing processes to restore the database and use the toolkit task to restore Solr. Regardless of the mechanism that you use to restore the components, you must restore the database before you restore the Solr index.

To deploy to a previous point in time, your deployment must be functional. The procedure assumes that Solr, ZooKeeper, and the database management system are running. As part of the process, you must stop the Liberty servers in the deployment. To recover a deployment after a disaster, see Recovering from a disaster.

In a deployment that contains the i2 Connect gateway only, you only need to restore the i2 Analyze and Liberty configuration.

For more information about the restore toolkit tasks, see:

- Back up and restore the database
- Back up and restore Solr
- Back up and restore the configuration

Run each toolkit command from a Liberty server in your deployment.

Procedure

1. Use the `validateBackups` to ensure that the timestamps of the backups you are restoring are compatible.
   This task compares the timestamps of the backups created by the toolkit to ensure that the configuration backup is the earliest, and that the Solr backup precedes the database backup.

   ```
   setup -t validateBackups -p configTimestamp=<timestamp> -p
    solrTimestamp=<timestamp> -p dbTimestamp=<timestamp>
   ```

   Where `configTimestamp` is the timestamp of a configuration backup, `solrTimestamp` is the timestamp of a Solr backup, and `dbTimestamp` is the timestamp of a database backup. You do not have to provide all of the timestamps to run the toolkit task.

   If the backups are compatible, continue with the restore process.
2. Before you restore the deployment, stop Liberty. In a deployment that contains multiple Liberty servers, you must run the `stopLiberty` command on every Liberty server.

   ```
   setup -t stopLiberty
   ```
3. To restore the database in your deployment, run the following command from a Liberty server:

   ```
   setup -t restoreDatabases -p timestamp=<timestamp>
   ```

   The database backup is identified in the location from the `backup.db.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup in the location.

   For example: `setup -t restoreDatabases -p timestamp=20210420145332`.
4. To restore the Solr search index, run the following command from a Liberty server:

   ```
   setup -t restoreSolr -p timestamp=<timestamp> --hostname
    <liberty.hostname>
   ```

   Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the host-name attribute of the `<application>` element in the topology.xml file.

The Solr backup is identified in the location from the `backup.solr.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup directory in the location.

For example: `setup -t restoreSolr -p timestamp=20210420145332 --hostname <liberty.hostname>`.

5. To restore the i2 Analyze and Liberty configuration, run the following command from a Liberty server:

   ```
   setup -t restoreConfiguration -p timestamp=<timestamp>
   ```

   The configuration backup is identified in the location from the `backup.config.location.dir` setting by using the timestamp that was specified in the command.
   The timestamp value in the command must match the timestamp of a backup directory in the location.
   In a deployment with high availability, run this command on each Liberty server.

   For example: `setup -t restoreConfiguration -p timestamp=20210420145332`.

6. Start Liberty.
   In a deployment that contains multiple Liberty servers, you must run the `startLiberty` command on every Liberty server.

   ```
   setup -t startLiberty
   ```

Result

Your deployment of i2 Analyze is restored to the point in time of the backups that you specified.

Recovering from a disaster

You can use the backups of the components of i2 Analyze to restore the system, or parts of a the system, after a disaster. In a production deployment of i2 Analyze, use a tested backup and restore procedure to ensure that you can recover from failures.

About this task

The following steps describe the process of restoring a deployment of i2 Analyze by using the provided toolkit tasks. You do not need to restore every component by using the toolkit tasks. For example, you can use existing processes to restore the database and use the toolkit task to restore Solr. Regardless of the mechanism that you use to restore the components, you must restore the components in the same order as the following steps.

For more information about the restore toolkit tasks, see:

- Back up and restore the database
- Back up and restore Solr
- Back up and restore the configuration

In a deployment that contains the i2 Connect gateway only, you only need to restore the i2 Analyze and Liberty configuration.

To recover from a disaster, you might need to reinstall the i2 Analyze deployment toolkit and install the components of i2 Analyze on the failed servers. The following process assumes that you are recreating the servers for each component in your deployment.

Procedure

1. Create the environment to restore your deployment in to. If your servers were lost due to a disaster, you might need to recreate them with the required prerequisites. For more information about the servers that are used in a deployment of i2 Analyze, see Deployment topologies.
2. Use the `validateBackups` to ensure that the timestamps of the backups you are restoring are compatible.
   This task compares the timestamps of the backups created by the toolkit to ensure that the configuration backup is the earliest, and that the Solr backup precedes the database backup.

   ```
   setup -t validateBackups -p configTimestamp=<timestamp> -p
    solrTimestamp=<timestamp> -p dbTimestamp=<timestamp>
   ```

   Where `configTimestamp` is the timestamp of a configuration backup, `solrTimestamp` is the timestamp of a Solr backup, and `dbTimestamp` is the timestamp of a database backup. You do not have to provide all of the timestamps to run the toolkit task.

   If the backups are compatible, continue with the restore process.
3. Restoring to new Liberty servers.
   a. Copy the `i2a-config-<timestamp>\toolkit-config` directory from your backup to the `toolkit\configuration` directory of the deployment toolkit on each server in your environment.

      You might have a different configuration backup for each Liberty server in your deployment, ensure that you copy the correct backup for each Liberty server.
4. Create a new deployment to restore into. To create a new deployment, follow the instructions to install, deploy, and start the components in Deploying i2 Analyze on multiple servers.

   By starting the components, you ensure that your environment is deployed successfully. You can now restore your environment from your backups.
5. Before you restore the deployment, stop Liberty. In a deployment that contains multiple Liberty servers, you must run the `stopLiberty` command on every Liberty server.

   ```
   setup -t stopLiberty
   ```
6. To restore the database in your deployment, run the following command from a Liberty server:

   ```
   setup -t restoreDatabases -p timestamp=<timestamp>
   ```

   The database backup is identified in the location from the `backup.db.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup in the location.

   For example: `setup -t restoreDatabases -p timestamp=20210420145332`.
7. To restore the Solr search index, run the following command from a Liberty server:

   ```
   setup -t restoreSolr -p timestamp=<timestamp> --hostname
    <liberty.hostname>
   ```

   Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the host-name attribute of the `<application>` element in the topology.xml file.

   The Solr backup is identified in the location from the `backup.solr.location.dir` setting by using the timestamp that was specified in the command. The timestamp value in the command must match the timestamp of a backup directory in the location.

For example: `setup -t restoreSolr -p timestamp=20210420145332 --hostname <liberty.hostname>` .

8. Deploying to new Liberty servers.
   a. To restore the i2 Analyze and Liberty configuration, run the following command from a Liberty server:

      ```
      setup -t restoreConfiguration -p timestamp=<timestamp>
      ```

      - The configuration backup is identified in the location from the `backup.config.location.dir` setting by using the timestamp that was specified in the command.
      - The timestamp value in the command must match the timestamp of a backup directory in the location.
      - In a deployment that contains multiple Liberty servers, run the `restoreConfiguration` command on each Liberty server.

      For example: `setup -t restoreConfiguration -p timestamp=20210420145332`.

   b. Start Liberty.
      In a deployment that contains multiple Liberty servers, you must run the `startLiberty` command on each Liberty server.

      ```
      setup -t startLiberty
      ```

Result

Your deployment of i2 Analyze is restored to new servers after a disaster.

Back up and restore the configuration

The i2 Analyze and Liberty configuration contains the files that define the functionality of the deployment, and how users connect to the deployment. To restore the deployment after a failure, create a backup of the configuration.

Intro

- The `configuration` directory and some files from the Liberty deployment are backed up. These files enable you to recreate a deployment from the backup files in the same configuration.
- The system can be running when you back up the configuration. The system must be stopped when you restore the configuration.
- In a distributed deployment of i2 Analyze, the configuration directory should be identical on each server. Therefore, you only need to back up the configuration from the Liberty server.

Backup location

In the `environment.properties` file, the `backup.config.location.dir` setting is the location where the configuration backups are created and restored from when you use the toolkit tasks. This location must exist on the Liberty server where you run the toolkit tasks.

Toolkit tasks

- `backupConfiguration`

- The `backupConfiguration` toolkit task backs up the i2 Analyze and Liberty configuration to the `backup.config.location.dir` location.
  For example:

```
setup -t backupConfiguration
```

- `restoreConfiguration`
  - The `restoreConfiguration` toolkit task restores the i2 Analyze and Liberty configuration from a backup. The `restoreConfiguration` also deploys Liberty. You must pass the `timestamp` parameter to the toolkit task with a timestamp for a backup in the `backup.config.location.dir` location.
    For example:

```
setup -t restoreConfiguration -p timestamp=<timestamp>
```

    If you are recovering from a disaster scenario where you must create new Liberty servers, you must copy the i2 Analyze deployment toolkit configuration from the backup to your new Liberty servers before you run the `restoreConfiguration` task.

Backup files

When the `backupConfiguration` task is run, a directory is created named `i2a-config-<timestamp>` where `<timestamp>` is the timestamp of when the toolkit task was run. The directory contains the files of the i2 Analyze and Liberty configuration and a file that shows the version of i2 Analyze. For example:

```
- i2a-config-20210420145332
    - version.txt
    - liberty-app-config
    - liberty-shared-config
    - toolkit-config
```

The following directories are included in the backup:

- `toolkit\configuration`
- `wlp\usr\shared\configs`
- `wlp\usr\servers\opal-server`

Back up and restore Solr

When you use the `backupSolr` and `restoreSolr` toolkit tasks to back up and restore the Solr index and ZooKeeper configuration, a call to the Solr REST API is made. The toolkit tasks create and restore a full backup of the non-transient Solr collections.

Introduction

- The non-transient Solr collections are backed up and restored by using the provided toolkit tasks. The following collection types are non-transient:
  - `main_index`
  - `match_index`
  - `chart_index`
- You must take the Solr backup before a database backup, and you must restore Solr after you restore the database.

Backup location

In the `environment.properties` file, the `backup.solr.location.dir` setting is the location where the Solr index and ZooKeeper configuration backups are created and restored from. This location must be accessible by every Solr node in your deployment.

Toolkit tasks

- `backupSolr`

  The `backupSolr` toolkit task backs up the Solr index and ZooKeeper configuration to the `backup.solr.location.dir` location.
  For example:

  ```
  setup -t backupSolr
  ```

- `restoreSolr`

  The `restoreSolr` toolkit task restores the Solr index and ZooKeeper configuration from a backup.

  You must pass the `timestamp` parameter to the toolkit task with a value for a backup in the `backup.solr.location.dir` location.

  You must also specify the `hostname` parameter, with the hostname of the Liberty server where you are running the command, and that matches the value for the host-name attribute of the `<application>` element in the topology.xml file. For example:

  ```
  setup -t restoreSolr -p timestamp=<timestamp> --hostname
   <liberty.hostname>
  ```

Backup files

When the `backupSolr` task is run, a directory is created named `i2a-solr-<collection_name>-<timestamp>` where `<timestamp>` is the timestamp of when the toolkit task was run and `<collection_name>` the name of the Solr collection backed up. A directory is created for each non-transient Solr collection, with each collection containing a backup. For example:

```
- i2a-solr-chart_index-20210420145332
    - backup.properties
    - snapshot.shard1
    - snapshot.shard2
    - zk_backup
- i2a-solr-main_index-20210420145332
    - backup.properties
    - snapshot.shard1
    - snapshot.shard2
    - zk_backup
- i2a-solr-match_index1-20210420145332
    - backup.properties
    - snapshot.shard1
    - snapshot.shard2
    - zk_backup
- i2a-solr-match_index2-20210420145332
    - backup.properties
    - snapshot.shard1
    - snapshot.shard2
    - zk_backup
```

Back up and restore the database

When you use the `backupDatabases` and `restoreDatabases` toolkit tasks to back up and restore the database in your deployment, a call to the database management system is made. The toolkit tasks create and restore a full backup of the database.

Intro

- The `backupDatabases` toolkit task completes a full backup of the database. The application must be stopped before you can back up the database and there must be no active connections to the database.
  - Refer to your database management system documentation to learn how to ensure that there are no connections to your database before you create the backup.
- The toolkit task is aimed for smaller deployments where the system can be offline while the database backup is completed. If you have a larger deployment or don't want the system to be offline at all, you can back up the database manually. To back up and restore the database manually refer to the documentation for your database management system:
  - For Db2, see Developing a backup and recovery strategy
  - For SQL Server, see Backup Overview
  - For PostgreSQL, see Backup and Restore
- If you back up the database manually, you can still use the toolkit tasks to back up the i2 Analyze configuration and the Solr index.
- When you back up a deployment of i2 Analyze, the database must be backed up after the Solr index. When you restore the database, you must restore the database before the Solr index.

Backup location

In the `environment.properties` file, the `backup.db.location.dir` setting is the location where the database backup are created and restored from. This location must be accessible from the database server. The user that is specified in the `credentials.properties` file for your database management system must have write permissions to this location.

Toolkit tasks

- `backupDatabases`

  The `backupDatabases` toolkit task backs up the database to the `backup.db.location.dir` location.
  For example:

  ```
  setup -t backupDatabases
  ```
- `restoreDatabases`

  The `restoreDatabases` toolkit task restores the database from a backup. You must pass the `timestamp` parameter to the toolkit task with a value for a backup in the `backup.db.location.dir` location.
  For example:

  ```
  setup -t restoreDatabases -p timestamp=<timestamp>
  ```

Backup files

A backup file is created in the location specified in `backup.db.location.dir`. The file name includes a timestamp of when the backup is created. For example:

- On Db2: `ISTORE.0.db2inst1.DBPART000.20210604144506.001`
- On SQL Server: `ISTORE.20210604152638.bak`
- On PostgreSQL: `ISTORE.20210604152638.dump`

You can verify the backup file was correctly created by running the following commands on your database server:

- On Db2: `db2ckbkp <backup file>`
- On SQL Server:

  ```
  RESTORE VERIFYONLY FROM DISK = "<backup file>";
  GO
  ```
- On PostgreSQL, there is no specific command for verifying a backup. Instead, you can restore the backup onto a different server. Alternatively, you can perform a quick check by using the `pg_restore` command to list the contents of a backup file without testing their integrity.

Administering high availability

When your system is deployed in an environment that is designed for high availability or disaster recovery, you must develop, configure, and test your continuous operation procedures. Continuous operation includes, but is not limited to, components of the application continuing to function after a subset of their resources is taken offline.

In summary, when a server, or number of servers, fail in your environment you need to ensure that the system continues to function. To do so, the process involves:

- Detecting server failure
- Ensuring that only the functional servers are used to process requests
- Recovering the failed servers
- Configuring the environment to use the recovered servers again

The implementation of this process is different for each component of i2 Analyze, the software prerequisites that are used, and the operating system that the component is deployed on. For some components of i2 Analyze, the process is automatic but you should understand the process and what state the environment is in.

The following information outlines the process for each component in a deployment of i2 Analyze.

Liberty

In a deployment that provides high availability, multiple Liberty servers are used to provide active/active availability. The active/active pattern allows multiple Liberty servers to allow connections from clients and continue to function when some of the Liberty servers fail.

Detecting server failure

To detect if a Liberty server has failed, you can use logs in your load balancer. In the load balancer, monitor the response where you use the `health/live` endpoint to determine which Liberty servers to route clients to. If the response from the `health/live` endpoint for a particular Liberty server is `503`, that Liberty server may have failed.

For more information about the endpoint and its responses, see The health/live endpoint.

The Liberty server can return a `503` if it is not started, in the startup process, or there is a temporary loss of connection to another component. Liberty can restart as a non-leader if it temporarily loses the connection to a component and loses leadership. If this is the reason for the `503`, the following message is displayed in the `i2_Component_Availability.log` file:

```
INFO  ApplicationStateHandler        - I2ANALYZE_STATUS:0066 - Application
 is entering non-leader mode to serve requests
```

For more information about the log file, see Monitor the system availability.

Automatic failover

If a Liberty server fails, the other Liberty servers continue to function as usual. If a client was connected to the failed Liberty server, the analyst might have to log out from the client and log in again for the load balancer to route the request to one of the live Liberty servers.

If it was the leader Liberty server that failed, one of the remaining Liberty servers is elected leader. For more information about the leadership process, see Liberty leadership configuration.

Recovering failed servers

There are a number of reasons why a server might fail. Use the logs from the failed server to diagnose and solve the issue. For example, you might need to restart the server, increase the hardware specification, or replace hardware components.

- The Liberty logs are in the `deploy\wlp\usr\servers\opal-server\logs` directory.
- The deployment toolkit logs are in the `toolkit\configuration\logs` directory.

For more information about the different log files and their contents, see Deployment log files.

Reinstating high availability

On the recovered Liberty server, run `setup -t startLiberty` to restart the server and i2 Analyze application.

You can use the load balancer logs to ensure that the Liberty server is now returning `200` from the `health/live` endpoint.

Liberty leadership configuration

In a deployment with multiple Liberty servers, one server is determined to be the leader.

Liberty leadership behavior

The leader Liberty can complete actions that require exclusive access to the Information Store database. These actions include;

- Initiating Solr indexing
- Initiating online upgrade tasks
- Initiating alerts for saved Visual Queries
- Initiating deletion-by-rule jobs

Leadership state

ZooKeeper is used to determine and manage the leadership state of the Liberty servers in a deployment. Each Liberty server polls ZooKeeper for the current leadership state. If there is no leader, the Liberty attempts to become the leader. One Liberty will successfully become the leader.

Leadership poll

To become a leader, and to maintain leadership, Liberty must be able to connect to ZooKeeper, Solr, and the database management system. The connection to each component is checked whenever the leadership state is polled. If any of the components cannot be connected to, the Liberty either releases its leadership or cannot run for leader if there is not currently a leader.

The leadership poll schedule is independent on each Liberty server. If the current leader releases its leadership, there is no leader until the leadership poll runs on another Liberty that can then assume leadership status.

The leadership poll is completed on a configurable schedule. By default, the leadership state is checked every five minutes. To change the interval, update the value for the `ServerLeadershipReconnectIntervalMinutes` setting in `DiscoServerSettingsCommon.properties`.

For example:

```
ServerLeadershipReconnectIntervalMinutes=5
```

Error thresholds

The leadership state can change if a certain number of errors are encountered during indexing operations within a specified time span. These types of operations are run only on the leader Liberty. The maximum number of errors that can occur, and the time span in which they can occur, are configurable.

The `MaxPermittedErrorsBeforeLeadershipRelease` setting defines the maximum number of errors that can occur in the time specified by the `TimeSpanForMaxPermittedErrorsBeforeLeadershipReleaseInSeconds` setting. By default, up to five errors can occur within 30 seconds before the Liberty relinquishes its leadership.

For example:

```
MaxPermittedErrorsBeforeLeadershipRelease=5
TimeSpanForMaxPermittedErrorsBeforeLeadershipReleaseInSeconds=30
```

Restarting limits

If the leader Liberty releases leadership, it attempts to restart in non-leader mode. If Liberty cannot connect to each component, it fails to start. By default, it tries to start three times before failing. To start Liberty on this server again, resolve the issue that is stopping Liberty connecting to all of the components and run `setup -t startLiberty`.

You can change the maximum number of times that Liberty attempts to start before it fails. To change the number of attempts, update the value of the `ServerLeadershipMaxFailedStarts` setting in `DiscoServerSettingsCommon.properties`.

```
ServerLeadershipMaxFailedStarts=3
```

The `health/live` endpoint

The `api/v1/health/live` REST endpoint returns the status of the i2 Analyze application running on a Liberty server.

The `api/v1/health/live` endpoint can return one of two possible responses:

- `200 The application is UP and handling requests.`
- `503 The application is DOWN and not handling requests.`

When the endpoint is called, the application checks whether it can connect to ZooKeeper, Solr, and the database management system if your deployment uses one. If it can connect to all of the components, the endpoint returns `200`. This means that you can route client requests to this Liberty server.

If the application fails to connect to a component, the endpoint returns `503`. The Liberty server cannot process client requests.

The endpoint also returns `503` until the application completes the start process.

When you configure a load balancer for use in a high availability deployment, it should use the `api/v1/health/live` endpoint to determine which servers to send requests to. For more information, see Deploying a load balancer.

Database management system

In a deployment that provides high availability, the Information Store database is deployed in an active/passive pattern. In the active/passive pattern, there is a primary database instance and a number of standbys.

Detecting server failure

There are a number of different ways to detect if there has been a database failure.

- The `i2_Component_Availability.log` contains messages that report the connection status for the Information Store database. For more information about the messages that are specific to the database, see *Database management system messages* in Monitor the system availability.

There are also a number of different tools that are specific to your database management system.

- Db2:
  - Db2 fault monitor on Linux
  - Heartbeat monitoring in clustered environments
  - Monitoring Db2 High Availability Disaster Recovery (HADR) databases

  For more information about detecting failures for Db2, see Detecting an unplanned outage.
- SQL Server
  - To understand when SQL Server initiates failover, see How Automatic Failover Works.

Manual and automatic failover

When the primary server fails, the system must fail over and use the remaining servers to complete operations.

- Db2
  - If you are using an automated-cluster controller, failover to a standby instance is automatic. For more information, see .

    - If you are using client-side automatic client rerouting, then you must manually force a standby instance to become the new primary. For more information about initiating a takeover, see Performing an HADR failover operation.
- SQL Server
    - When SQL Server is configured for high availability in an availability group with three servers, failover is automatic. For more information about failover, see Automatic Failover.

Recovering failed servers

There are a number of reasons why a server might fail. You can use the logs from the database server to diagnose and solve the issue. For example, you might need to restart the server, increase the hardware specification, or replace hardware components that caused the issue.

When the server is back online and functional, you can recover it to become the primary server again. Alternatively, you can recover it to become the standby for the new primary that you failed over to.

This might include recovering a back up of the Information Store database from before the server failed.

Reinstating high availability

- Db2
    - Some toolkit tasks only work on the original primary database server when you are not using an automated-cluster controller such as TSAMP. To use those toolkit tasks, you must revert to using the original primary database server after a failure or redeploy your system to use the new primary database server.
        - For more information about the process to make the recovered database the primary again, see Reintegrating a database after a takeover operation.
        - To redeploy with the new primary, update the `topology.xml` on each Liberty to reference the new server in the `host-name` and `port-number` and redeploy each Liberty.
- SQL Server
    - On SQL Server it is not required to return to the previous primary, however you might choose to do so. You can initiate a planned manual failover to return the initial primary server. For more information, see Planned Manual Failover (Without Data Loss).

Solr

In a deployment that provides high availability, i2 Analyze uses SolrCloud to deploy Solr with fault tolerance and high availability capabilities. Solr uses an active/active pattern that allows all Solr servers to respond to requests.

Detecting server failure

To detect if there has been a Solr failure, you can use the following mechanisms:

- The `i2_Component_Availability.log` contains messages that report the status of the Solr cluster. For more information about the messages that are specific to Solr, see the *Solr messages* section in Monitor the system availability.
- The Solr Web UI displays the status of any collections in a Solr cluster. Navigate to the following URL in a web browser to access the UI: `http://localhost:8983/solr/#/~cloud?view=graph`. Where `localhost` and `8983` are replaced with the hostname and port number of one of your Solr nodes.

Automatic failover

Solr continues to update the index and provide search results while there is at least one replica available for each shard.

Recovering failed servers

There are a number of reasons why a server might fail. Use the logs from the server to diagnose and solve the issue.

Solr logs:

- The Solr logs are located in the `i2analyze\deploy\solr\server\logs\8983` directory on each Solr server. Where `8983` is the port number of the Solr node on the server.
- The application logs are in the `deploy\wlp\usr\servers\opal-server\logs` directory, specifically the `i2_Solr.log` file.

  For more information about the different log files and their contents, see Deployment log files.

To recover the failed server you might need to restart the server, increase the hardware specification, or replace hardware components.

Reinstating high availability

On the recovered Solr server, run `setup -t startSolrNodes -hn <solr.hostname>` to start the Solr nodes on the server.

You can us the `i2_Component_Availability.log` and Solr Web UI to ensure that the nodes start correctly and that the cluster returns to its previous state.

ZooKeeper

In a deployment that provides high availability, ZooKeeper is used to manage the Solr cluster and maintain Liberty leadership information. ZooKeeper uses an active/active pattern that allows all ZooKeeper servers to respond to requests.

Detecting server failure

To detect ZooKeeper server failures:

- The `i2_Component_Availability.log` contains messages that report the status of the ZooKeeper quorum. For more information, see the *ZooKeeper status messages* section in Monitor the system availability.
- The Solr Web UI contains the status of the ZooKeeper quorum: `http://localhost:8983/solr/#/~cloud?view=zkstatus`. Where `localhost` and `8983` are replaced with the host name and port number of one of your Solr nodes.

Manual and automatic fail over

The ZooKeeper quorum continues to function while more than half of the total number of ZooKeeper hosts in the quorum are available. For example, if you have three ZooKeeper servers, your system can sustain one ZooKeeper server failure.

Recovering failed servers

There are a number of reasons why a server might fail. Use the logs from the server to diagnose and solve the issue.

ZooKeeper logs:

- The ZooKeeper logs are located in the `i2analyze\data\zookeeper\8\ensembles\zoo\zkhosts\<1>\logs` directory on the failed ZooKeeper server. Where `1` is the identifier of the ZooKeeper host on the server.
- The application logs are in the `deploy\wlp\usr\servers\opal-server\logs` directory.

For more information about the different log files and their contents, see Deployment log files.

To recover the failed server you might need to restart the server, increase the hardware specification, or replace hardware components.

Reinstating high availability

On the recovered ZooKeeper server, run `setup -t startZkHosts -hn <zookeeper.hostname>` to start the ZooKeeper host on the server.

You can use the `i2_Component_Availability.log` and Solr Web UI ZooKeeper status view to ensure that the system returns to its previous state.

Monitor the system availability

i2 Analyze reports on the availability of each component in the environment so that the state can be monitored. The availability is logged on each Liberty server in the environment.

To monitor the status of your deployment, use the `i2_Component_Availability.log` file. This log file is located in the `wlp/servers/opal-server/opal-services/logs` directory on each Liberty server in your deployment. Therefore, you must monitor the log file on each Liberty server in your deployment.

The log contains messages about the Liberty leadership status, Solr cluster status, and the availability of each component.

Liberty leadership messages

If the Liberty starting the election process, the following message is displayed:

```
INFO  ServerLeadershipMonitor        - We are running for Liberty leader
```

If the Liberty is elected leader, the following messages are displayed:

```
INFO  ServerLeadershipMonitor        - We are the Liberty leader
INFO  ApplicationStateHandler        - I2ANALYZE_STATUS:0065 - Application
 is entering leader mode
```

If the Liberty is not elected leader, the following messages are displayed:

```
INFO  ServerLeadershipMonitor        - We are not the Liberty leader
INFO  ApplicationStateHandler        - I2ANALYZE_STATUS:0063 - Application
 is entering non-leader mode to serve requests
```

Component messages

If all of the components are available to the Liberty server, the following message is displayed:

```
INFO  ComponentAvailabilityCheck     - All components are available
```

If at least one of the components in not available to the Liberty server, the following message is displayed:

```
WARN  ComponentAvailabilityCheck     - Not all components are availale
```

When the Liberty server continues to check the availability of each component, the following message is displayed:

```
INFO  ComponentAvailabilityCheck     - I2ANALYZE_STATUS:0068 - The
 application is waiting for all components to be available
```

Solr messages

If the Liberty server cannot connect to the Solr cluster, the following message is displayed:

```
WARN  ComponentAvailabilityCheck     - Unable to connect to Solr cluster
WARN  ComponentAvailabilityCheck     - The Solr cluster is unavailable
```

The following messages describe the state of the Solr cluster in the deployment:

- ALL_REPLICAS_ACTIVE - The named Solr collection is healthy. All replicas in the collection are active.

  For example:

  ```
  INFO  SolrHealthStatusLogger         - 'main_index','ALL_REPLICAS_ACTIVE'
  ```

- DEGRADED - The named Solr collection is degraded. The minimum replication factor can be achieved, but at least one replica is down or failed to recover.

  For example:

  ```
  WARN  SolrHealthStatusLogger         - 'main_index','DEGRADED'
  ```

  When the status is `DEGRADED`, data can still be written to the Solr index. When the status returns to `ALL_REPLICAS_ACTIVE`, the data is synchronized in the Solr index as described in Data synchronization in i2 Analyze.

  The deployment can still be used with the collection in a degraded state, however the deployment can now sustain fewer Solr server failures. If a degraded state is common, or lasts for an extended time, you should investigate the Solr logs to improve the stability of the system.

- RECOVERING - The named Solr collection is recovering. The minimum replication factor cannot be achieved, because too many replicas are currently in recovery mode.

  For example:

  ```
  INFO  SolrHealthStatusLogger         - 'main_index','RECOVERING'
  ```

  If all of the replicas recover, the status changes to `ALL_REPLICAS_ACTIVE`. If the replicas fail to recover, the status changes to `DEGRADED` or `DOWN`.

- DOWN - The named Solr collection is down. The minimum replication factor cannot be achieved because too many replicas are down or have failed to recover.

  For example:

  ```
  WARN  SolrHealthStatusLogger         - 'main_index','DOWN'
  ```

  When the status is `DOWN`, data cannot be written to the Solr index. You should attempt to resolve the issue. For more information, see Solr.

- UNAVAILABLE - The named Solr collection is unavailable. The application cannot connect to the collection.

  For example:

  ```
  WARN  SolrHealthStatusLogger         - 'main_index','UNAVAILABLE'
  ```

  When the status is `UNAVAILABLE`, data cannot be written to the Solr index. You should attempt to resolve the issue. For more information, see Solr.

ZooKeeper status messages

If the connection to ZooKeeper is lost, the following messages are displayed:

```
WARN  ComponentAvailabilityCheck     - Unable to connect to ZooKeeper
WARN  ComponentAvailabilityCheck     - The ZooKeeper quorum is unavailable
```

When the connection to the database is restored, the all components are active message is displayed.

Database management system messages

If the connection to the database is lost, the following messages are displayed:

```
WARN  ComponentAvailabilityCheck     - Information Store database appears to
 be offline, retrying...
WARN  ComponentAvailabilityCheck     - Unable to connect to the Information
 Store database.
java.sql.SQLException: The TCP/IP connection to the host has failed.
Error: "Socket operation on nonsocket: configureBlocking. Verify the
 connection properties.
Make sure that an instance of SQL Server is running on the host and
 accepting TCP/IP connections at the port.
Make sure that TCP connections to the port are not blocked by a firewall.".
 DSRA0010E: SQL State = 08S01, Error Code = 0
WARN  ComponentAvailabilityCheck     - The Information Store database is
 unavailable
```

When the connection to the database is restored, the all components are active message is displayed.

Data synchronization in i2 Analyze

In a deployment of i2 Analyze that contains an Information Store database, data is stored in both the database and the Solr search index. In i2 Analyze, the database is considered the source of truth for data in the deployment. When data is added, modified, or removed from the deployment, the data change is applied to the database before those changes are reflected in the search index.

When data changes occur, the database and Solr retain watermarks that record if the operation was successfully completed in the database and Solr index. i2 Analyze routinely checks these watermarks in the deployment. i2 Analyze can respond to the state of the deployment in a number of different ways, and can normally be resolved without manual intervention.

Note: There is no mechanism in i2 Analyze to ensure that data is written to any standby database instances before the watermarks are updated.

Changes to the Solr collection health

If at least on replica for a shard is unavailable, the collection is marked as unhealthy. If there are data changes in the database during this time, the Solr index is updated on the replicas that are available while the minimum replication factor is achieved.

When all of the replicas are available and the Solr collection is marked as healthy again, all of the data changes that occurred when it was in the unhealthy state are reindexed. This means that all data changes are indexed on every replica.

At i2 Analyze application start

When the i2 Analyze application starts, i2 Analyze checks the watermarks.

- If the deployment is in sync, then the deployment starts as usual.

- If there is data in the database that isn't in the Solr index, the indexing process indexes the data from the database that is not in the Solr index.
- If there is data in the Solr index that isn't in the database, i2 Analyze does not start and the following message is displayed:

  ```
  The 'main_index' index reports that it is ahead of the database
  ```

  This situation can occur if you recover the database from a backup that was taken before the current Solr index. In your back up and restore procedure, ensure that you restore the database from a backup that was taken after the Solr index backup.

Implementation details

The values that i2 Analyze uses to record the state of the deployment are called *watermarks*. Whenever data is changed in the database, the data change is then indexed in Solr. When Solr has completed indexing the changed data, the watermarks are updated. The watermarks are stored in the database and ZooKeeper. Two watermarks are maintained, a *high watermark* and a *low watermark*.

High watermark

The high watermark is updated in both locations when Solr achieves the minimum replication factor. That is, when the index is updated on a specified number of replicas for a shard. When you configure Solr for high availability, you specify the minimum replication factor for each Solr collection.

Low watermark

The low watermark is updated in both locations when Solr achieves the maximum replication factor. That is, when the index is updated on all of the replicas for a shard. The low watermark is only updated after the high watermark, the low watermark cannot have a higher value than the high watermark.

Configuration resources

The configuration resources section contains information that is referenced elsewhere in the configuration section. The section also includes the reference documentation for the configuration files that are in the i2 Analyze deployment toolkit.

Configuration files reference

In the deployment toolkit, you can identify and modify the core components of your deployment. Many of the settings in the configuration files must be set before you can deploy i2 Analyze.

Specifying the deployment credentials

To allow the deployment toolkit to update the database, Lightweight Third-Party Authentication (LTPA) keys, and Solr search platform components, you must provide user names and passwords. The user names and passwords that you provide allow the system to set up and administer components of i2 Analyze, and are not used to access i2 Analyze.

About this task

Database

For the database that is identified in `topology.xml`, you must specify a user name and a password in the `credentials.properties` file. The `setup` script uses this information to authenticate with the database.

Note: The user that you specify must have privileges to create and populate databases in the database management system.

The database credentials are stored in the following format:

```
db.infostore.user-name=<user name>
db.infostore.password=<password>
```

For example:

```
db.infostore.user-name=admin
db.infostore.password=password
```

The `db.infostore.truststore.password` credential is used only when you configure the connection between the database and Liberty to use SSL. If you are not using SSL to secure this connection, you do not need to specify a value for the `db.infostore.truststore.password` credential. For more information about configuring SSL, see Configure Secure Sockets Layer with i2 Analyze.

If you change this password after deployment, the change will take effect when you redeploy the system.

LTPA keys

You must provide a value for the `ltpakeys.password` property. This value is used by the system to encrypt LTPA tokens.

- For a stand-alone deployment of i2 Analyze, you can specify any value as the password.
- For a deployment of i2 Analyze that uses LTPA tokens to authenticate with other systems, you must specify the same password that those systems use.

If you change this password after deployment, you must run the `setup -t clearLTPAkeys` toolkit task before the change will take effect when you redeploy the system.

Solr search platform

The Solr search platform is used to search data in the Information Store. You must provide values for the `solr.user-name` and `solr.password` properties. Any Solr indexes are created when i2 Analyze is first deployed, and the values that you provide here become the Solr user name and password.

If you have already deployed i2 Analyze, and you want to change the Solr password, new properties must be created in the following format:

```
solr.user-name.new=value
solr.password.new=value
```

For example:

```
solr.user-name=admin
solr.password={enc}E3FGHjYUI2A\=

solr.user-name.new=admin1
solr.password.new=password
```

After you provide the new values, run the `setup -t configureSolrSecurity` toolkit task. You must also run `setup -t deployLiberty` on your Liberty server, and `setup -t createSolrNodes` on your Solr servers. As a part of the deployment process, the new values replace the original values in the file.

When you deploy i2 Analyze, the passwords in the `credentials.properties` file are encoded.

Procedure

1. Using a text editor, open the `toolkit\configuration\environment\credentials.properties` file.
2. Specify the database, LTPA, and Solr credentials for your deployment:
   - the database user names and passwords
   - the LTPA keys password
   - the Solr user name and password
3. Save and close the `credentials.properties` file.

The `environment.properties` file

The following properties are in the `environment.properties` file:

Property	Description	Default value
db.installation.dir	The installation path of the database management system or the database management system client.	If you are using IBM Db2, `C:/Program Files/IBM/SQLLIB` on Windows, or `/opt/ibm/db2/V11.5` on Linux.
		If you are using Microsoft SQL Server, `C:/Program Files/Microsoft SQL Server/Client SDK/ODBC/170` on Windows, or `/opt/mssql-tools` on Linux.
		If you are using PostgreSQL, `C:/Program Files/PostgreSQL/16` on Windows, or `/usr/lib/postgresql/16` on Linux (or `/usr/pgsql-16` on RHEL).
db.database.location.dir	The root path of the database file storage.	If you are using IBM Db2, `C:` on Windows, or `/home/db2inst1` on Linux.
		If you are using Microsoft SQL Server, `C:/Program Files/Microsoft SQL Server/MSSQL16.MSSQLSERVER/MSSQL/DATA` on Windows, or `/var/opt/mssql/data` on Linux.
		If you are using PostgreSQL, `C:/Program Files/PostgreSQL/16/data` on

Property	Description	Default value
		Windows, or `/var/lib/postgresql/data` on Linux (or `/var/lib/pgsql/16/data` on RHEL).
wlp.home.dir	The installation path for Open Liberty.	`C:/i2/i2analyze/deploy/wlp` on Windows, or `/opt/i2/i2analyze/deploy/wlp` on Linux.
solr.home.dir	The installation path for Apache Solr.	`C:/i2/i2analyze/deploy/solr` on Windows or `/opt/i2/i2analyze/deploy/solr` on Linux.
zookeeper.home.dir	The installation path for Apache ZooKeeper.	`C:/i2/i2analyze/deploy/zookeeper` on Windows or `/opt/i2/i2analyze/deploy/zookeeper` on Linux.
apollo.data	The path to a directory where i2 Analyze can store files.	`C:/i2/i2analyze/data` on Windows, or `/opt/i2/i2analyze/data` on Linux.
backup.config.location.dir	The location where the i2 Analyze configuration backups are created and restored from. This location must exist on the Liberty server where you run the backupConfiguration command.	
backup.solr.location.dir	The location where the Solr index and ZooKeeper configuration backups are created and restored from. This location must be accessible by every Solr node in your deployment.	
backup.db.location.dir	The location where the database backup are created and restored from. The user that is specified in the credentials.properties file for your database management system must have write permissions to this location.	

The `topology.xml` file

In the deployment toolkit, the topology file defines the physical architecture of the finished deployment, and the components that make up the pieces of that architecture. The topology file contains domain-specific information that you must provide before deployment can take place.

The `topology.xml` file contains the information that the `setup` script requires to identify the prerequisites that are being used, where the components of the system are located, and the fragments to combine to create the application.

The `topology.xml` file contains distinct sections for defining different aspects of a deployment.

`i2-data-sources`

A definition of each data source that is available within the deployment. For more information, see Data sources.

`databases`

The names, types, and locations of the databases that i2 Analyze uses to store data. For more information, see Databases.

`applications`

The applications that comprise this deployment of i2 Analyze, and the locations of the application servers on which they are to be installed. For more information, see Applications.

`zookeepers` and `solr-clusters`

The ZooKeeper hosts and Solr clusters that are used in this deployment. For more information, see Solr and ZooKeeper.

`connectors`

The connectors that are used with the i2 Connect gateway in this deployment. For more information, see Connectors.

After you update the topology file, you can either modify other aspects of the deployment toolkit or redeploy the system to update the deployment with any changes. For more information about redeploying your system, see Redeploying Liberty.

Data sources

The topology files that are supplied with each example in the deployment toolkit contains a preconfigured `<i2-data-source>` element.

The `<i2-data-sources>` element contains a single `<i2-data-source>` element for the data source that the deployment connects to. For example:

```
<i2-data-sources>
    <i2-data-source default="false" id="infostore">
        <DataSource Version="0" Id="">
            <Name>Information Store</Name>
        </DataSource>
     </i2-data-source>
</i2-data-sources>
```

Where:

Attribute	Description
`id`	A unique identifier that is used to distinguish this data source throughout the system.
`default`	An optional attribute. The value must be `false` in this version of i2 Analyze.

Each `<i2-data-source>` contains a single `<DataSource>` element that has two standard attributes and two child elements.

Attribute	Description
`Id`	Reserved for future use. The value must be empty in this version of i2 Analyze.
`Version`	Reserved for future use. The value must be `0` in this version of i2 Analyze.

Element	Description
`Name`	The name of this data source, which is presented to users.

Databases

The `<databases>` element defines the database that i2 Analyze uses to store data. The `<database>` element contains attributes that define information about the database, and the mechanism that is used to connect to it.

For example:

```
<database database-type="InfoStore"
dialect="db2" database-name="ISTORE" instance-name="DB2"
xa="false" host-name="<host>" id="infostore"
port-number="50000" />
```

Where:

Attribute	Description
`database-type`	The value must be `InfoStore` in this version of i2 Analyze.
`dialect`	Specifies the type of database engine. This attribute can be set to one of `postgres`, `db2`, or `sqlserver`.
`database-name`	A name that identifies the database to the database engine.

Attribute	Description
`instance-name`	The instance name that was specified during installation of your database engine. If you are using SQL Server, you can either specify the `port-number` or `instance-name` attribute. For SQL Server on Linux, you must use the `port-number` attribute. This attribute is not applicable to PostgreSQL databases.
`xa`	Determines whether distributed transactions are enabled for this database.
`host-name`	The hostname of the server where the database is located.
`id`	A unique identifier that is used to distinguish this database throughout the system. The value must match the `id` in the `<i2-data-source>` element.
`port-number`	The port on the database server to send requests to. If you are using SQL Server, you can either specify the `port-number` or `instance-name` attribute.

Remote database attributes

If your database is on a remote server from the i2 Analyze application, you must use the following attributes in the `topology.xml` file:

Attribute	Description
`os-type`	Identifies the operating system of the server where the database is located. This attribute can be set to one of `WIN` or `UNIX`. If you are using Db2, you can also specify `AIX`.
`node-name`	Identifies the node to create in the Db2 node directory. The value of the `node-name` attribute must start with a letter, and have fewer than 8 characters. For more information about naming in Db2, see Naming conventions. This attribute is not applicable to PostgreSQL or SQL Server databases.

For example, if you are using PostgreSQL:

```
<database database-type="InfoStore" dialect="postgres"
    database-name="ISTORE" xa="false" id="infostore"
    host-name="<remote.hostname>" port-number="5432" os-type="WIN" />
```

For example, if you are using IBM Db2:

```
<database database-type="InfoStore" dialect="db2" instance-name="DB2"
    database-name="ISTORE" xa="false" id="infostore"
    host-name="<remote.hostname>" port-number="50000"
    node-name="node1" os-type="WIN" />
```

For example, if you are using Microsoft SQL Server:

```
<database database-type="InfoStore" dialect="sqlserver" instance-
name="SQLSERVER"
    database-name="ISTORE" xa="false" id="infostore"
    host-name="<remote.hostname>" os-type="WIN" />
```

Applications

The `<applications>` and child `<application>` element within the topology file define the indexes, file stores, and WAR file for the application.

An `<application>` element has the following attributes:

Attribute	Description
`name`	The value must be `opal-server` in this version of i2 Analyze
`host-name`	The hostname of the server where the application is located.
`http-server-host`	Determines whether the HTTP server is configured by the deployment toolkit. Set to `true` to configure the HTTP server, or `false` not to configure the HTTP server.

The following sections explain how to define the indexes, file stores, and WAR file for an application.

File stores

The `<application>` element contains a child `<file-stores>` element that defines the file stores that are used by the application. The `location` attribute specifies the file path to use for each file store. The other attributes must be left with their default values.

WAR files

The `<application>` element contains a child `<wars>` element. The `<wars>` element contains child `<war>` elements that define the contents of the i2 Analyze WAR files that are installed on the application server.

Each `<war>` element has the following attributes:

Attribute	Description
`target`	The type of WAR file to create. The following types are available: `opal-services-is`, `opal-services-daod`, `opal-services-is-daod`.

Attribute	Description
name	The name of the WAR file.
i2-data-source-id	A name that identifies the <i2-data-source> element for this WAR file.
context-root	The URL that is used to access the WAR file. For the opal-services-is and opal-services-is-daod WARs, this is opal by default. For the opal-services-daod WAR, this is opaldaod by default.

The following child elements of the <war> element can be defined:

<data-sources>

Identifies the database that is used by this WAR in a child <data-source> element. The <data-source> element has the following attributes:

Attribute	Description
database-id	Identifies the database to use. This value must match the id value that is specified in a <database> element. For the opal-services-is and opal-services-is-daod WARs, the <data-source> element must reference a database of type InfoStore.
create-database	Determines whether the database is to be created on the server that is specified in the <database> element. Set as true to create the database, or false not to create the database.

<solr-collection-ids>

Identifies the Solr collections that are used by this WAR in child <solr-collection-id> elements. Each Solr collection is identified by the collection-id attribute, the value of which must match the id value that is specified in a <solr-collection> element.

A Solr collection belongs to a Solr cluster. Each Solr cluster is identified by the cluster-id attribute, the value of which must match an id value that is specified in a <solr-cluster> element.

For more information about the ZooKeeper and Solr-related elements in the topology.xml file, see Solr and ZooKeeper.

<file-store-ids>

Identifies the file stores that are used by this WAR. Each file store is identified by the value attribute, the value of which must match an id value that is specified in a <file-store> element.

<connector-ids>

Identifies the connectors that are available in the `opal-services-is-daod` and `opal-services-daod` WARs in a child `<connector-id>` element. Each connector is identified by the `value` attribute, the value of which must match an `id` value that is specified in a `<connector>` element.

For more information about connector-related elements in the `topology.xml` file, see Connectors.

`<fragments>`

Specifies the fragments that are combined to create the WAR.

Note: All WARs must contain the `common` fragment.

In the example `topology.xml` file for the `information-store-opal` example deployment, the `opal-server` application definition is:

```
<application name="opal-server" host-name="" http-server-host="true">
  <wars>
    <war context-root="opal" name="opal-services-is"
         i2-data-source-id="infostore" target="opal-services-is">
      <data-sources>
        <data-source database-id="infostore" create-database="true" />
      </data-sources>
      <file-store-ids>
        <file-store-id value="chart-store" />
        <file-store-id value="job-store" />
        <file-store-id value="recordgroup-store" />
      </file-store-ids>
      <fragments>
        <fragment name="opal-services-is" />
        <fragment name="opal-services" />
        <fragment name="common" />
        <fragment name="default-user-profile-provider" />
      </fragments>
      <solr-collection-ids>
        <solr-collection-id collection-id="main_index" cluster-
id="is_cluster" />
        <solr-collection-id collection-id="match_index1" cluster-
id="is_cluster" />
        <solr-collection-id collection-id="match_index2" cluster-
id="is_cluster" />
        <solr-collection-id collection-id="highlight_index" cluster-
id="is_cluster" />
        <solr-collection-id collection-id="chart_index" cluster-
id="is_cluster" />
        <solr-collection-id collection-id="vq_index" cluster-
id="is_cluster" />
        <solr-collection-id collection-id="recordshare_index" cluster-
id="is_cluster" />
      </solr-collection-ids>
    </war>
  </wars>
  <file-stores>
    <file-store location="" id="job-store" type="job-store" />
    <file-store location="" id="recordgroup-store" type="recordgroup-
store" />
  </file-stores>
</application>
```

In the example `topology.xml` file for the `daod-opal` deployment, the `opal-server` application definition is:

```
<application http-server-host="false" name="opal-server" host-name="">
```

```
  <wars>
    <war context-root="opaldaod" name="opal-services-daod"
         i2-data-source-id="opalDAOD" target="opal-services-daod">
      <file-store-ids>
        <file-store-id value="recordgroup-daod-store" />
      </file-store-ids>
      <fragments>
        <fragment name="opal-services" />
        <fragment name="common" />
      </fragments>
      <solr-collection-ids>
        <solr-collection-id collection-id="daod_index" cluster-
id="is_cluster"/>
      </solr-collection-ids>
    </war>
  </wars>
  <file-stores>
    <file-store location="" id="recordgroup-daod-store" type="recordgroup-
store" />
  </file-stores>
</application>
```

Solr and ZooKeeper

i2 Analyze uses Solr for text indexing and search capabilities. The ZooKeeper service maintains configuration information and distributed synchronization across Solr and Liberty.

The `topology.xml` file for a deployment that includes the `opal-server` application also includes the `<zookeepers>` and `<solr-clusters>` elements. The `<solr-clusters>` and `<zookeepers>` elements define the Solr cluster that is used in a deployment and the ZooKeeper instance that manages it.

`<solr-clusters>`

In the supplied `topology.xml` file that includes the `opal-server` application with the `opal-services-is` WAR, the `<solr-clusters>` definition is:

```
<solr-clusters>
  <solr-cluster id="is_cluster" zookeeper-id="zoo">
    <solr-collections>
      <solr-collection
        id="main_index" type="main"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="match_index1" type="match"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="match_index2" type="match"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="highlight_index" type="highlight"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
```

```
      <solr-collection
        id="chart_index" type="chart"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="vq_index" type="vq"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
      <solr-collection
        id="recordshare_index" type="recordshare"
        lucene-match-version=""
        num-shards="4" num-replicas="1"
      />
    </solr-collections>
    <solr-nodes>
      <solr-node
        memory="2g"
        id="node1"
        host-name=""
        data-dir=""
        port-number="8983"
      />
    </solr-nodes>
  </solr-cluster>
</solr-clusters>
```

The `<solr-clusters>` element includes a child `<solr-cluster>` element. The `id` attribute of the `<solr-cluster>` element is a unique identifier for the Solr cluster. To associate the Solr cluster with the ZooKeeper instance, the value of the `zookeeper-id` attribute must match the value of the `id` attribute of the `<zookeeper>` element.

`<solr-collections>`

The `<solr-collections>` element is a child of the `<solr-cluster>` element. The `<solr-collections>` element has child `<solr-collection>` elements.

Depending on the WARs that are included in the application, the number and type of required child `<solr-collection>` elements is different.

- `opal-services-is`

  In the `opal-services-is` WAR, you must have `<solr-collection>` elements with each of the following values for the `type` attribute:

  - `main` - you must have either one or two collections of type `main`
  - `match` - you must have two collections of type `match`
  - `highlight` - you must have one collection of type `highlight`
  - `chart` - you must have one collection of type `chart`
  - `vq` - you must have one collection of type `vq`
  - `recordshare` - you must have one collection of type `recordshare`
- `opal-services-daod`

  In the `opal-services-daod` WAR, you must have one `<solr-collection>` element with a value of `daod` for the `type` attribute.
- `opal-services-is-daod`

In the `opal-services-is-daod` WAR, you must have `<solr-collection>` elements with each of the following values for the `type` attribute:

- `main` - you must have either one or two collections of type `main`
- `daod` - you must have one collection of type `daod`
- `match` - you must have two collections of type `match`
- `highlight` - you must have one collection of type `highlight`
- `chart` - you must have one collection of type `chart`
- `vq` - you must have one collection of type `vq`
- `recordshare` - you must have one collection of type `recordshare`

The `<solr-collection>` element has the following attributes:

Attribute	**Description**
`id`	An identifier that is used to identify the Solr collection.
`type`	The type of the collection. The possible values are: `main`, `daod`, `match`, `highlight`, `chart`, `vq`, `recordshare`.
`lucene-match-version`	The version of the Lucene matching behavior for the collection. At this release, the value is populated when you deploy i2 Analyze.
`num-shards`	The number of logical shards that are created as part of the Solr collection.
`num-replicas`	The number of physical replicas that are created for each logical shard in the Solr collection.
`min-replication-factor`	The minimum number of replicas that an update must be replicated to for the operation to succeed. This optional value must be greater than 0 and less than or equal to the value of `num-replicas`.
`num-csv-write-threads`	The number of threads that are used to read from the database and write to the temporary CSV file when indexing data in the Information Store. This attribute is optional, and applies to Solr collections of type `main` and `match` only. The total of `num-csv-write-threads` and `num-csv-read-threads` must be less than the number of cores available on the Liberty server.
`num-csv-read-threads`	The number of threads that are used to read from the temporary CSV file and write to the index when indexing data in the Information Store. This attribute is optional, and applies to Solr collections

Attribute	Description
	of type `main` and `match` only. This value must be less than the value of `num-shards`. The total of `num-csv-write-threads` and `num-csv-read-threads` must be less than the number of cores available on the Liberty server.

<solr-nodes>

The <solr-nodes> element is a child of the <solr-cluster> element. The <solr-nodes> element can have one or more child <solr-node> elements. Each <solr-node> element has the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the Solr node.
memory	The amount of memory that can be used by the Solr node.
host-name	The hostname of the Solr node.
data-dir	The location where Solr stores the index.
port-number	The port number of the Solr node.

<zookeepers>

In the supplied `topology.xml` file that includes the `opal-server` application, the <zookeepers> definition is:

```
<zookeepers>
  <zookeeper id="zoo">
    <zkhosts>
      <zkhost
        id="1"
        host-name=""
        data-dir=""
        port-number="9983"
        quorum-port-number=""
        leader-port-number=""
      />
    </zkhosts>
  </zookeeper>
</zookeepers>
```

The <zookeepers> element includes a child <zookeeper> element. The `id` attribute of the <zookeeper> element is a unique identifier for the ZooKeeper instance. To associate the ZooKeeper instance with the Solr cluster, the value of the `id` attribute must match the value of the `zookeeper-id` attribute of the <solr-cluster> element.

The <zkhosts> element is a child of the <zookeeper> element. The <zkhosts> element can have one or more child <zkhost> elements. Each <zkhost> element has the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the ZooKeeper host. This value must be an integer in the range 1 - 255.
host-name	The hostname of the ZooKeeper host.
data-dir	The location that ZooKeeper uses to store data.
port-number	The port number of the ZooKeeper host.
quorum-port-number	The port number that is used for ZooKeeper quorum communication. By default, the value is 10483.
leader-port-number	The port number that is used by ZooKeeper for leader election communication. By default, the value is 10983.

Connectors

An i2 Analyze deployment that includes the i2 Connect gateway enables you to connect to external data sources.

The `topology.xml` file for a deployment that includes the `opal-server` application with the `opal-services-is-daod` or `opal-services-daod` WARs includes the `<connectors>` element. The `<connectors>` element defines the connectors that are used in a deployment.

`<connectors>`

The `<connectors>` element contains one or more child `<connector>` elements. The `<connector>` element can have the following attributes:

Attribute	Description
id	A unique identifier that is used to identify the connector.
name	The name of the connector, which is presented to users.
base-url	The URL to the connector, made up of host name and port number. For example, `https://<host name>:<port number>`. You can use the HTTP or HTTPS protocol.
configuration-url	The URL to any configuration that is required for the connector. The default value for the `configuration-url` attribute is `/config`. The presence of the attribute in the `topology.xml`

Attribute	Description
	file is optional. If it is not present, the default value is used.
`gateway-schema`	The short name of the gateway schema whose item types the connector can use, overriding any setting in the connector configuration. This attribute is optional; when present its value must match one of the settings in `ApolloServerSettingsMandatory.properties`.
`schema-short-name`	The short name by which the connector schema for the connector is known in the rest of the deployment, overriding any setting in the connector configuration. Use this optional attribute to avoid naming collisions, or to change the name that users see in client software.

After a successful deployment, the `topology.xml` file in the `daod-opal`, `chart-storage-daod`, and `information-store-daod-opal` example configurations contains the following `<connectors>` definition:

```
<connectors>
  <connector id="example-connector" name="Example"
             base-url="http://localhost:3700/" />
</connector>
```

The `ApolloServerSettingsMandatory.properties` file

The following properties are in the `ApolloServerSettingsMandatory.properties` file:

Property	Description	Default value
SchemaResource	The file that contains the schema for the Chart Store or the Information Store. This setting can be removed if the deployment includes neither store. When specified, this XML file needs to be on the classpath.	`law-enforcement-schema.xml`
ChartingSchemesResource	The file that contains charting schemes for the Chart Store or the Information Store. This setting can be removed if the deployment includes neither store. When specified, this XML file needs to be on the classpath.	`law-enforcement-schema-charting-schemes.xml`
Gateway.External.SchemaResource	A file that contains a gateway schema for this deployment of	

Property	Description	Default value
	i2 Analyze. This setting can be removed if the deployment does not require a gateway schema. When specified, this XML file needs to be on the classpath.	
Gateway.External.ChartingSchemesFileResource	A file that contains gateway charting schemes for this deployment of i2 Analyze. This setting can be removed if the deployment does not include a gateway schema. When specified, this XML file needs to be on the classpath.	
DynamicSecuritySchemaResource	The security schema file, which defines the security dimensions and values that are available in the system, and the mapping of user groups to dimension values. This (XML) file needs to be on the classpath.	`example-dynamic-security-schema.xml`
AuditLogger	The full class name of the audit logger. If no auditing is required, specify `com.i2group.disco.audit.NoOpAuditLogger`	`com.i2group.disco.audit.NoOpAuditLog`
DefaultSecurityDimensionValuesProvider	The full class name of the default security dimension values provider. Use `com.i2group.disco.user.PropertyBasedDefaultSecurityDimensionV` to set the default security dimension values on a record to the list from the DefaultSecurityDimensionValues element in your security schema. This is only applicable to the Information Store with Opal Services.	`com.i2group.disco.user.PropertyBased`
UserGroupsProvider	The full class name of the user groups provider, when i2 Analyze is in just-in-time provisioning mode. To retrieve user groups from the Open Liberty user registry, specify `com.i2group.disco.user.WebSphereUserGroupsProvider` To retrieve user groups from	`com.i2group.disco.user.WebSphereUser`

Property	Description	Default value
	a claims-based mechanism, for example OpenID Connect (OIDC), specify `com.i2group.disco.user.ClaimsBasedUserGroupsProvider`	
UserIdentityCompatibilityMode	Since version 4.4.4, i2 Analyze supports using externally provided unique identifiers to identify users. Earlier versions always used principal names for user identification. When you upgrade from version 4.4.3 or earlier, existing data that identifies users by their principal names cannot support a change to user identifiers. During upgrade, the installer enables a compatibility mode that allows the system to continue to use principal names, by setting UserIdentityCompatibilityMode to 'true'.	`false`
ProvisioningCompatibilityMode	Since version 4.4.4, i2 Analyze supports a mechanism for provisioning users and groups from a file when the application starts, creating an "allow list" for those who are permitted to use the application. When you upgrade from version 4.4.3 or earlier, existing configuration for federated user registries might not produce the correct results. During upgrade, the installer enables a compatibility mode in which users and groups are provisioned "just-in-time" as they authenticate, by setting ProvisioningCompatibilityMode to 'true'.	`false`

The `DiscoServerSettingsCommon.properties` file

The following properties are in the `DiscoServerSettingsCommon.properties` file:

Property	Description	Default value
IdleLogoutMinutes	The idle time in minutes after which the end user is logged out. The value of this setting must be greater than AlertBeforeIdleLogoutMinutes.	`15`
AlertBeforeIdleLogoutMinutes	The time in minutes the end user is alerted prior to being logged out due to inactivity. The value of this setting must be less than IdleLogoutMinutes.	`2`
AlwaysAllowLogout	Forces logout to always be available for all authentication methods.	`false`
ResultsConfigurationResource	The file that specifies what options are available to users when they view and filter results.	
CommandAccessControlResource	The file that is used to specify specific group access to commands.	
EnableSolrIndexScheduler	Turn on or off the scheduling of indexing Setting this option to false disables the scheduler and should be used when ingesting large amounts of data.	`true`
SolrHealthCheckIntervalInSeconds	The interval time in seconds between the checks of the Solr cluster status.	`60`
QueryTimeoutSeconds	The time in seconds after which the server can cancel a search, resulting in an error. This is not an absolute limit. A search might continue to run for several seconds after the limit is reached, but it should terminate within a reasonable time. A zero setting disables search timeout.	`60`

Property	Description	Default value
WildcardMinCharsWithAsterisk	The minimum number of characters (not counting asterisks) that must be provided in a wildcard text search query that contains asterisks.	`0`
WildcardMinCharsWithQuestionMark	The minimum number of characters (not counting question marks or asterisks) that must be provided in a wildcard text search query that contains question marks.	`0`
SearchTermMaxLevenshteinDistance	The maximum Levenshtein distance for spelled-like text searches. The allowed values are 2, 1, and 0. Set the value to 0 to turn off spelled-like text searches.	`2`
VisualQueryTimeoutMinutes	The time in minutes after which the server cancels a running visual query and sends an error response. Set the value to 0 to turn off visual query timeouts.	`240`
VisualQueryWildcardMinCharsWithAsterisk	The minimum number of characters (not counting asterisks) that must be provided in a visual query condition that contains or implies asterisks. (Matches wildcard pattern; Starts with; Ends with; Contains)	`0`
VisualQueryWildcardMinCharsWithQuestionMark	The minimum number of characters (not counting question marks or asterisks) that must be provided in a visual query condition that contains question marks. (Matches wildcard pattern)	`0`
VisualQueryConfigurationResource	The file that specifies what operators are valid in visual query conditions that involve particular property types of particular item types.	

Property	Description	Default value
VisualQueryMaxValuesInList	The maximum number of values in the value list of a visual query condition.	`10000`
RecordMaxNotes	The maximum number of notes that can be added to a record.	`50`
MaxRecordsPerDeleteRequest	The maximum number of records that can be deleted in one request.	`500`
MaxSourceIdentifiersPerRecord	The maximum number of source identifiers that can be present on a record.	`50`
MaxSeedsForDaodServices	The maximum number of records to send as seeds to any i2 Connect service in a "DAOD" or "combined" deployment.	`500`
AlertScheduleTimeOfDay	The time of day to run the Visual Queries that are saved with alerting enabled. The format is HH:mm, HH is the hour of the day in a 24-hour time format, 00-23, and mm is the number of minutes past the hour, 00-59.	`00:00`
ExpandMaxResultsSoftLimit	The soft limit for the maximum number of results that can be returned for an Expand operation.	`0`
SourceReferenceSchemaResource	The file that restricts the source names and types that users can specify in source references when creating and importing records.	
LinkMatchEndRecordCombinationLimit	The maximum number of link end combinations that are searched for when performing link matching.	`100`
ChartUnsavedChangesLifespanDays	The length of time for which the server retains unsaved changes to a chart across user sessions.	`7`

Property	Description	Default value
	Any change to an unsaved chart resets the timer.	
ChartUnsavedChangesCleanUpSchedule	The schedule on which to clean up unsaved changes to charts when ChartUnsavedChangesLifespanDays is exceeded. The format is a Unix C cron expression.	`0 0 * * 0`
I2ConnectRecordMatchAction	The action to perform when records in an i2 Connect result set match other records in the set. The permitted values for this setting are IGNORE, UNITE, and MERGE.	`IGNORE`
EnableWelcomePage	Enables a welcome page that is displayed when a user first connects from Analyst's Notebook 10.	`false`
EnablePrivacyPrompt	Enables a privacy agreement that users must accept during login in order to access the i2 Analyze server.	`false`
PrivacyAcceptancePeriodDays	The period after which a user who has accepted the privacy agreement must read and accept it again. When this property has no value, users do not see the agreement again until the mechanism is reset.	
EnableMetrics	Enables the generation of server metrics for the consumption of monitoring software such as Prometheus.	`false`
SecurityDimensionValuesReloadInterval	The schedule on which to reload information from any specified security dimension values provider. These settings have an effect only when a provider class is specified in the security schema. Reasonable values for	`15`

Property	Description	Default value
	the units setting are MINUTES, HOURS, and DAYS.	
SecurityDimensionValuesReloadIntervalUnits		`MINUTES`
SecurityDimensionValuesProviderTimeout	A timeout on the duration of calls to a dimension values provider. If a call doesn't return within the configured period, an exception is thrown. Reasonable values for the units setting are SECONDS and MINUTES.	`60`
SecurityDimensionValuesProviderTimeoutUnits		`SECONDS`
SecurityPermissionsProviderTimeout	A timeout on the duration of calls to the permissions provider. If a call doesn't return within the configured period, an exception is thrown. Reasonable values for the units setting are SECONDS and MINUTES.	`60`
SecurityPermissionsProviderTimeoutUnits		`SECONDS`
MaxChartletSizeInMB	The maximum size in megabytes of a chartlet that a user creates when they share records. Chartlets are stored in Solr, so raising this limit might require extra Solr storage resources.	`100`
ChartletLifespan	The lifespan of a chartlet determines how long it remains in the index before it becomes eligible for deletion. The lifespan is a combination of magnitude and units. The units can be SECONDS, MINUTES, HOURS, or DAYS.	`7`
ChartletLifespanUnits		`DAYS`
DeploymentDisplayName	The display name for the deployment, which might appear in a client user interface.	

Redeploying and resetting i2 Analyze

Most changes to the configuration of i2 Analyze require you to redeploy the system. During development, it is common to clear data from Information Store, or to remove the Information Store and re-create it.

About this task

Significant changes to the Information Store schema or the security schema of an i2 Analyze deployment usually invalidate the information in the Information Store, or even the store itself. As you develop your schemas, you often need either to remove and re-ingest your test data, or to remove and re-create the store.

After any change to the toolkit configuration of a deployment of i2 Analyze, you must redeploy in order to update to the running deployment.

Performing these actions in sequence is common before in your development environments, but rare in production. If you do run them in sequence, the combination effectively resets the system to its just-deployed state.

You might also need to reset a deployment of i2 Analyze to its just-installed state. For example, if you completed an example deployment and then want to start developing a production deployment on the same hardware.

Clearing data from the system

The `clearData` task can remove data that is stored in both the search index and the Information Store database. You can use this task to remove test data from a system.

About this task

In a multiple server deployment, run the `clearData` and `clearSearchIndex` tasks from the Liberty server.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `clearData` and `clearSearchIndex` on the Liberty server that you used to create the database only.

Important: These instructions are designed to remove data that is used for test purposes. Do not use the task to clear data in a production system without backing up your data. For more information about backing up your data, see Backing up a deployment.

Procedure

1. On the Liberty server, in a command prompt navigate to the `toolkit\scripts` directory.
2. Stop the deployment:

   ```
   setup -t stopLiberty
   ```
3. To clear data and the search index, run the following command:

   ```
   setup -t clearData --hostname <liberty.host-name>
   ```

   - Where `liberty.hostname` is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.
   - A message is displayed when you run the task to confirm that you want to complete the action. Enter `Y` to continue. The data and the search index are removed from the system.

If you run the `clearData` task with the `--scripts` argument, the scripts for clearing the Information Store database are generated in the `toolkit\scripts\database\<database dialect>\InfoStore\generated\clearData` directory, but not run. You can then inspect the scripts before they are run, or run the scripts yourself. To run the scripts yourself, run them in their numbered order to clear the data from your system.

To generate the scripts for the task, run:

```
setup -t clearData --scripts
```

After you run the scripts manually, clear the search index:

```
setup -t clearSearchIndex --hostname <liberty.host-name>
```

4. Start the deployment of i2 Analyze.

```
setup -t startLiberty
```

Resetting the system

The `dropDatabases` task can remove not only the data, but also the underlying structures such as tables from your database management system. The `deleteSolrCollections` task can remove any Solr collections from the system.

Before you begin

To drop the databases, you must ensure that there are no active connections to them.

About this task

In a multiple-server environment, run all toolkit tasks from the Liberty server.

In a deployment that provides high availability, use the documentation from your database management system to remove the Information Store database from each database server in your deployment instead of the `dropDatabases` toolkit task.

In a deployment that provides high availability, stop and start each Liberty server in your environment but run `clearSearchIndex` on the Liberty server that you used to create the database only.

Important: These instructions are designed to remove databases that are used for test purposes. Do not use the task to remove databases in a production system without backing up your data. For more information about backing up your data, see Backing up a deployment.

Procedure

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop the deployment:

```
setup -t stopLiberty
```

3. To remove the database and Solr collections, navigate to the `toolkit\scripts` directory and run the following command:

```
setup -t dropDatabases --hostname <liberty.host-name>
setup -t deleteSolrCollections --hostname <liberty.host-name>
```

Here, `liberty.hostname` is the hostname of the Liberty server where you are running the command. It matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

A message is displayed when you run each task to confirm that you want to complete the action. Enter Y to continue. The database and Solr collections are removed from the system.

4. To re-create the Solr collections and databases, run the following commands:

```
setup -t createSolrCollections --hostname <liberty.host-name>
setup -t createDatabases
```

5. Start the deployment of i2 Analyze.

```
setup -t startLiberty
```

Stopping and starting i2 Analyze

You might need to stop and start components of your i2 Analyze deployment to complete some configuration changes.

About this task

If you are using the single-server or remote database deployment topologies, you can use the `stop` and `start` toolkit tasks to stop and start i2 Analyze.

If you are using the multiple-server deployment topology, you must stop and start the components of i2 Analyze individually.

Procedure

1. Stop the components of i2 Analyze.
   a. Stop Liberty and the i2 Analyze application.

      To stop Liberty, run the following command on each Liberty server:

      ```
      setup -t stopLiberty
      ```

   b. Stop Solr.

      To stop Solr, run the following command on every server where Solr is running:

      ```
      setup -t stopSolrNodes --hostname <solr.host-name>
      ```

      Here, `solr.host-name` is the hostname of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

   c. Stop ZooKeeper.

      To stop ZooKeeper, run the following command on every server where ZooKeeper is running:

      ```
      setup -t stopZkHosts --hostname <zookeeper.host-name>
      ```

      Here, `zookeeper.host-name` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the `topology.xml` file.

2. Start the components of i2 Analyze.
   a. Start ZooKeeper.

      To start ZooKeeper, run the following command on every server where your ZooKeeper hosts are located:

      ```
      setup -t startZkHosts --hostname <zookeeper.host-name>
      ```

Here, `zookeeper.host-name` is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the `topology.xml` file.

b. Start Solr.

To start Solr, run the following command on every server where your Solr nodes are located:

```
setup -t startSolrNodes --hostname <solr.host-name>
```

Here, `solr.host-name` is the hostname of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

c. Start Liberty and the i2 Analyze application.

To start Liberty, run the following command on each Liberty server:

```
setup -t startLiberty
```

Redeploying Liberty

Redeploy Liberty to update the i2 Analyze application with your configuration changes.

About this task

In a multiple-server environment, run all toolkit tasks from the Liberty server.

If you follow this procedure in a deployment that provides high availability, you must complete each step on every Liberty server in your environment before you move to the next step.

Procedure

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Stop Liberty:
   ```
   setup -t stopLiberty
   ```
3. Update the i2 Analyze application:
   ```
   setup -t deployLiberty
   ```
4. Start Liberty:
   ```
   setup -t startLiberty
   ```
5. If you are using the IBM HTTP Server, start or restart it.

Setting up your XSD-aware XML editor

Some configuration files are provided with an associated XSD file. The XSD file provides client-side validation, and makes it easier to develop your XML configuration files.

About this task

It is recommended that you use an XSD-aware XML editor with intelligent code completion.

The following instructions describe how to install Visual Studio Code and the XML Language Support by Red Hat extension.

Procedure

1. Download and install Visual Studio Code, https://code.visualstudio.com/.
2. In Visual Studio Code, install the XML Language Support by Red Hat extension. For information about how to install extensions, see https://code.visualstudio.com/docs/editor/extension-gallery.

Results

After you install and configure your editor, you should use this editor whenever the instructions tell you to use an XSD-aware XML editor.

If you are using Visual Studio Code, the associated XSD file must be in the same directory as the XML file that you are editing.

Using the admin endpoints

Many of the admin endpoints in the REST API support the POST method, which you must call through a command-line tool such as `postman` or `curl`. *All* the admin endpoints require authentication, which means that you must log in to the server as an administrator and retrieve a cookie before you can call them.

Before you begin

To use the admin endpoints, you must log in as a user that has the necessary permission. For example, to provide a user with the `i2:Administrator` permission, you must use command access control to allocate it to a group that they're a member of. For more information, see Configuring command access control.

About this task

The following steps demonstrate how to use `curl` to POST to an admin endpoint.

You can find the REST API reference documentation at the following URL on the i2 Analyze server: `http://<host_name>/<context_root>/doc`. For example, `http://localhost/opal/doc`.

Procedure

1. If the `curl` utility is not available on your server, download it from the project website at https://curl.haxx.se/download.html.
2. Open a command prompt and use `curl` to log in to the i2 Analyze server and retrieve an authorization cookie:

   ```
   curl -i --cookie-jar cookie.txt
        -d j_username=user_name
        -d j_password=password
        http://<host_name>/<context_root>/j_security_check
   ```

   This command connects to the specified i2 Analyze server as the specified user, retrieves the authentication cookie, and saves it to a local file named `cookie.txt`. The LTPA token in the cookie is valid for 2 hours.

   When the command completes, the response from the request is displayed. If the retrieval is successful, the first line of the response is:

   ```
   HTTP/1.1 302 Found
   ```

After you retrieve and save the cookie, you can POST to an admin endpoint. When you do so, include the cookie file that you received when you logged in.

1. The following command is an example of a POST to the admin endpoint for reloading the live configuration:

   ```
   curl -i --cookie cookie.txt
        -X POST
        http://<host_name>/<context_root>/api/v1/admin/config/reload
   ```

   When the command completes, the response from the request is displayed. If the retrieval is successful, the first line of the response is:

   ```
   HTTP/1.1 200 OK
   ```

   If you see the `HTTP/1.1 403 Forbidden` response code, then either the user for which you retrieved the token does not have the necessary command access control permission, or the token has expired:

   - If your token has expired, you can get a new one by running the command in Step 2 again.
   - If your user does not have the `i2:Administrator` permission, see Configuring command access control to provide the user with the permission.

   Note: The `reload` method updates the configuration without requiring a server restart, but logged-in users might experience a short period of disruption when you run it.

Connecting to external data sources

Subject to your licensing terms, a deployment of i2® Analyze can use the i2 Connect gateway to query and retrieve data from external data sources. By connecting to an external data source, you enable a deployment of i2 Analyze to create and display records that represent the data in that source.

If the deployment and their permissions support it, users who create records from external sources can upload them to the Information Store for storage, sharing, and subsequent analysis.

Solutions that use the i2 Connect gateway depend on *connectors* that make the bridge between i2 Analyze and external data sources. Conventionally, you have one connector for each source that you connect to. Each connector provides one or more *services* that present a querying interface to users and perform the actual queries on the data source.

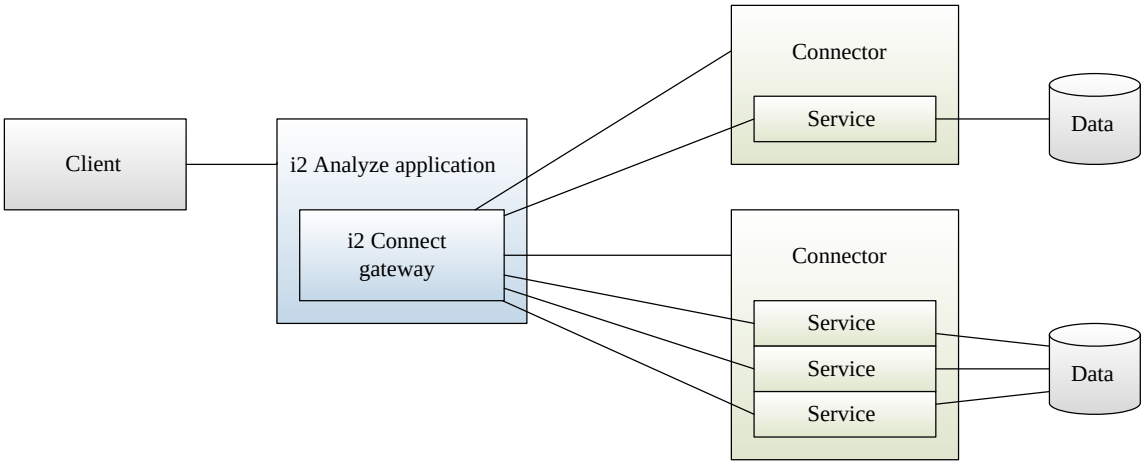
Note: i2 provides two ways for software developers to *create* connectors between the i2 Connect gateway and external data sources:

- The i2 Connect gateway defines a REST SPI that developers can implement. When the gateway calls SPI methods in response to user requests, an implementation can retrieve data from an external source and return it to the gateway. For information about this approach, and example connectors written in Java and Python, see the open-source project at https://i2group.github.io/analyze-connect
- The open-source i2 Connect SDK enables projects that include a *connector server*, which simplifies the connector development process by replacing the REST SPI and many configuration settings with a JavaScript/TypeScript API. In most situations, you should use the i2 Connect SDK.

i2 Analyze and the i2 Connect gateway

In an i2 Analyze deployment, you can provide users with access to an Information Store, or enable them to query data in one or more external sources, or both. i2® Analyze deployments that are targeted at external data access include the i2 Connect gateway alongside or instead of the Information Store.

In a complete solution, the i2 Connect gateway provides i2 Analyze with information about what connectors and services are available and how clients are to use them. i2 Analyze passes client requests for data to the gateway, which receives and responds to the requests on behalf of those services.



Each connector contains the code that interacts with an external data source, the definitions of the services that it supports, and the descriptions of how clients might present its services to users. For example, clients need to know the following information:

- The name and description of each service a user has access to
- Whether a service requires users to authenticate before they can use it
- Whether a service supports parameters that users can specify when they run it
- The input controls to display, and the validation to perform, for each parameter
- Whether a service behaves differently as a result of the current chart selection (any selected records are then *seeds* for the operation that the service performs)

The examples that i2 provides in the open-source repositories at https://i2group.github.io/analyze-connect and https://i2group.github.io/analyze-connect-node-sdk demonstrate meeting several of these requirements in fully functional connectors.

Note: For information about securing the communication between the i2 Connect gateway and its connectors, see Client authenticated Secure Sockets Layer and the i2 Connect gateway.

System security with the i2 Connect gateway

Three separate mechanisms govern different aspects of security in a deployment of i2 Analyze with the i2 Connect gateway. You can secure the connection between the gateway and a connector; you

can restrict which i2 Analyze users can run the queries that connectors implement; and connectors themselves can require users to authenticate before they can run queries.

- **Secure the connection to a connector**

  You can use client-authenticated SSL communication to secure the connection between the i2 Connect gateway and a connector. For more information, see Client-authenticated Secure Sockets Layer with i2 Connect.

- **Restrict access to a connector**

  You can use the command access control feature to control which users can use a connector. Through user groups, you can prevent users from using the i2 Connect gateway altogether, or you can restrict them to using only a subset of the deployed connectors. For more information, see Controlling access to features.

- **Require users to authenticate before running queries**

  Developers can write a connector so that users must authenticate before they can use one or more of the services that the connector provides. This feature enables connectors to interact with data sources that require users to log in, or to request authentication for their own purposes.

As well as these specific applications of i2 Analyze security, there are two other mechanisms that can affect which services a user has access to:

- When a user is prevented from seeing all records of a particular type, any service that requires seeds or returns results of that type is not visible to that user.
- A connector that supports user-specific configuration can, if it chooses, provide a different list of available services to different users.

Adding a connector to the topology

To add a connector to a deployment of i2 Analyze that includes the i2 Connect gateway, you must provide information about it in the topology file. In that file, the `<connector>` element controls the behavior of the connector and the experience of its users.

Before you begin

Ensure that your deployment of i2 Analyze includes the i2 Connect gateway. If the gateway was not a part of the original deployment, you can add it by following the instructions in Adding more data sources.

About this task

Each `<connector>` element in the topology file ties together the URL of the server that hosts the connector, and the URL of the configuration endpoint for that connector. After you change the contents of the topology file, you must update the deployment and restart the i2 Analyze server.

Procedure

When you deploy i2 Analyze with one of the example configurations that includes the i2® Connect gateway, the topology file contains a `<connectors>` element. Adding a connector to the topology means adding a child to that element. (If your topology file does not contain a `<connectors>` element, you can also add it by hand.)

1. Navigate to the `configuration\environment` directory of the toolkit for your deployment of i2 Analyze. In a text editor, open the `topology.xml` file.
2. In the `<war>` element, locate the `<connector-ids>` child element, or add it if necessary.

3. Add a `<connector-id>` child element with the following format:

```
<connector-ids>
  <connector-id value="ConnectorId"/>
</connector-ids>
```

   Here, `ConnectorId` must be unique within the topology file. Its purpose is to reference the configuration for the connector in the `<connectors>` element.

4. Add a `<connector>` element to the `<connectors>` element, which is a child of the root `<topology>` element:

```
<connectors>
  <connector id="ConnectorId"
             name="ConnectorName"
             base-url="Protocol://HostName:PortNumber"
             configuration-url="Protocol://HostName:PortNumber/Path"/>
</connectors>
```

   Here, `ConnectorName` is likely to be displayed to users in the list of queries that they can run against external sources.

   The i2 Connect gateway uses the value that you assign to the `base-url` attribute to locate the server that hosts the connector. `configuration-url` is the only optional attribute.

   By default, the gateway retrieves initial configuration information for the connector from `<base-url>/config`. You can change this behavior by specifying a different URL as the value of `configuration-url`.

   Important: Connectors created to target i2 Analyze 4.4.0 and above provide a `version` field in their configuration. This field enables a connector to tell the server which version of the SPI it requires. However, the new field causes a problem for a server that does not expect it.

   To support using an otherwise-compatible connector on a older i2 Analyze server, you must add `AllowUnknownConnectorConfigProperties=true` to any settings file.

5. If the connector requires the gateway to send headers that contain user information, add the `send-sensitive-headers` attribute to the `<connector>` element and set its value to `true`:

```
<connectors>
  <connector id="ConnectorId"
             name="ConnectorName"
             base-url="Protocol://HostName:PortNumber"
             configuration-url="Protocol://HostName:PortNumber/Path"
             send-sensitive-headers="true"/>
</connectors>
```

6. Save the file, and then restart the i2 Analyze server.

   a. On the command line, navigate to `toolkit\scripts`.

   b. Run the following commands in sequence:

```
setup -t stopLiberty
setup -t updateConnectorsConfiguration
setup -t startLiberty
```

   When the server restarts, the i2 Connect gateway makes its request to the configuration endpoint. If that endpoint is not available, the result is not an unrecoverable error. Rather, the i2 Analyze server logs the problem, and users see messages about unavailable services in the client application.

   Some connectors provide services that make use of definitions from the i2 Semantic Type Library. If you deploy a connector that was developed against a newer version of the library than the one on the server, compatibility problems can occur.

If you see a server startup message that mentions semantic types, you must update the server before you can use the connector. For more information, see the Troubleshooting guide in the i2 Connect gateway documentation.

To preview what users see, and to discover the status of the configured connectors at any time, you can use the server admin console.

1. Open a web browser and connect to the i2 Analyze server admin console at `http://host_name/opal/admin`. The admin console displays a list of connectors with their statuses. To see what the logged-in user would see in their client application, click **Preview services**.

Adding Connector Designer to your deployment

You can add Connector Designer to your existing deployment of i2 Analyze. Connector Designer runs on a container and allows users to create connectors that can be used to access external data sources.

Before you begin

You do not have to install Connector Designer on the same servers as your i2 Analyze deployment, but it must be able to communicate with the Liberty servers in your deployment. When you install Connector Designer, a load balancer is configured to route traffic to the i2 Analyze deployment and to Connector Designer. Before you start the process, ensure that you have configured the DNS for the load balancer and that you have the required certificates for the load balancer and the Connector Designer server.

For more information about the network and security architecture of Connector Designer, see Connector Designer deployment architecture.

The prerequisites for Connector Designer are as follows:

- x64 Linux
  - Connector Designer runs on a container and requires the Docker engine to be installed on an x64 Linux system.
- Docker Engine
  - To install the Docker Engine, see Install Docker Engine.
    - You can make the Docker engine available through Docker Desktop. Commercial use of Docker Desktop might require a paid subscription, Docker Core Subscriptions.

To add Connector Designer to your deployment, your existing deployment of i2 Analyze must be configured to use TLS when communicating with connectors. For more information, see Client-authenticated Transport Layer Security with the i2 Connect gateway

About this task

The following procedure describes how to add Connector Designer to your existing deployment of i2 Analyze. The process involves installing `analyze-deployment-tooling` which is used to install and run Connector Designer, configuring it to connect to your existing deployment, and configuring Connector Designer itself.

You can deploy Connector Designer in a development environment to let users create and test connectors before moving your Connector Designer configuration and connectors into your production environment.

Procedure

1. Download the `analyze-deployment-tooling.tar.gz` artifact from analyze-deployment-tooling on GitHub.
2. Open a terminal.
3. Navigate to the directory where you copied the downloaded file.
4. Extract the contents of the downloaded file.

   For example, `tar -xzf analyze-deployment-tooling.tar.gz`.
5. Navigate to the `analyze-deployment-tooling` directory.

   For example, `cd analyze-deployment-tooling`.
6. Download the i2 Analyze V4.4.4 Minimal for Linux.
   - To download i2 Analyze, follow the procedure described in Where can I download the latest i2 Products?
   - Populate the subject of the form with `Request for i2 Analyze 4.4.4 minimal toolkit for Linux`.
7. Rename the `i2analyzeMinimal_4.4.4.tar.gz` file to `i2analyzeMinimal.tar.gz`, then copy it to the `analyze-deployment-tooling/pre-reqs` directory.
8. Run the following command to install Connector Designer:

   `install-connector-designer`

   When you are prompted, provide the following information:
   - The fully qualified domain name (FQDN) for the load balancer. For example, `i2.my-organization`.
     - This is the FQDN that users will use to access the i2 Analyze application via the load balancer that is configured as part of installing and running Connector Designer. You must use the common name of the certificate that you are using for the load balancer.
   - Enter the port for the application on the load balancer. For example, `9443`.
   - The context root of your i2 Analyze application. By default this is `opal`.
   - The URLs of the Liberty servers in your i2 Analyze deployment. For example `https://liberty1:9082,https://liberty2:9082`.
9. Accept the license agreement.
   a. Read the notices file in `analyze-deployment-tooling/licenses/connector-designer/NOTICES`.
   b. If you agree to the terms, open the `analyze-deployment-tooling/licenses.conf` file and set `LIC_AGREEMENT=ACCEPT` and save the file.
10. To configure the TLS connection between the Connector Designer server and your existing i2 Analyze deployment, you must provide Connector Designer with the CA trust certificate for certificates that are received from Liberty. The certificate must be in PEM format and the certificate must be in the `analyze-deployment-tooling/environment-secrets/generated-secrets/certificates/externalCA/CA.cer` file. Overwrite the existing `CA.cer` file with your certificate.
11. You must obtain a signed certificate for the machine where you are deploying Connector Designer. You must provide the certificate and private key in the `environment-secrets/generated-secrets/certificates/i2analyze/server.key` and `environment-secrets/`

generated-secrets/certificates/i2analyze/server.cer files. Overwrite the existing server.key and server.cer files with your own.

The following steps must be completed in your i2 Analyze configuration:

1. Configuring JWT authentication.
   a. Copy the jwt-key.p12 and jwt-trust.p12 files from the environment-secrets/generated-secrets/certificates/jwt directory to your i2 Analyze deployment.
   b. Configure Liberty to use JWT tokens to allow authenticated users to access Connector Designer. For more information, see Configuring JWT authentication.

   - For the jwtKeyStore and jwtTrustStore elements, use the location of the jwt-key.p12 and jwt-trust.p12 files that you copied.
   - For the jwtKeyStore and jwtTrustStore elements, use the password in the environment-secrets/generated-secrets/certificates/jwt/jwt_PASSWORD file.
   - For the Issue=<i2-analyze-url> attributes, use the FQDN and port of the load balancer specified when you installed connector designer. For example, https://acme:9443.
2. In the i2 Analyze configuration, specify the Connector Designer configuration endpoint. In an XML editor, open the configuration/environment/topology.xml file and add the following element:

   ```
   <connector-config-providers>
       <connector-config-provider url="https://<fqdn>:<port>/connector-
   designer/api/configuration/connectors"/>
   </connector-config-providers>
   ```

   - Where <fqdn> is the FQDN of the load balancer and <port> is the port number that you specified when you installed Connector Designer. For example, https://i2.my-organization:9443.
3. The system requires an administrator user to be available that can access the i2 Analyze application. The user has the name adt-admin and be a member of the Administrator group. The password for the users is in the environment-secrets/generated-secrets/simulated-secret-store/adt-admin-password file.
4. Configure i2 Analyze for connection to Connector Designer:
   a. Edit the i2analyze/deploy/wlp/usr/servers/opal-server/jvm.options file to add the following line:

      ```
      -DBASIC_AUTH_ENDPOINTS=/api/v1/metrics /api/v1/gateway/reload /api/v1/
      health/live /api/v1/admin/*
      ```

   b. Add the adt-admin user and adt-admin-group to the i2analyze/deploy/wlp/usr/shared/config/user.registry.xml file:

      ```
      <user name="adt-admin" password="<password>"/>
      <group name="adt-admin-group">
      <member name="adt-admin"/>
      </group>
      ```

      - Where <password> is the password in the analyze-deployment-tooling/environment-secrets/generated-secrets/application/admin_PASSWORD file where you installed Connector Designer.

   c. Update the command access control file:

      ```
      <CommandAccessPermissions UserGroup="adt-admin-group">
          <Permission Value="i2:Administrator" />
      ```

```
</CommandAccessPermissions>
```

d. After you make these changes, redeploy and restart i2 Analyze. In a command prompt, navigate to the `toolkit/scripts` directory and run the following commands:

```
./setup -t deploy
./setup -t start
```

5. On the Connector Designer server, run the following command to start Connector Designer:

```
./scripts/deploy
```

6. To access the system, the machine that you are connecting from must trust the certificate that it receives from the load balancer. If the machine does not already trust the certificates. Install the `/environment-secrets/generated-secrets/certificates/externalCA/CA.cer` certificate as a trusted root certificate authority in your browser and operating system's certificate store.

   For information about installing the certificate, see:

   - Windows: Install Certificates with the Microsoft Management Console
   - FireFox:
     - In the settings menu, type **View Certificates** and open it. Then, click **Import** and locate the `CA.cer` file.

To connect to Connector Designer, log in to your deployment in a web browser at `https://<fqdn>:<port_number>/<context_root>`. For example, `https://i2.my-organization:9443/opal`.

Then, navigate to `https://<fqdn>:<port_number>/connector-designer/connectors`. For example, `https://i2.my-organization:9443/connector-designer/connectors`.

Configuring Connector Designer

1. To allow connectors to access external data sources, you must specify the allowed hosts in the `analyze-deployment-tooling/configs/con-des-default/configuration/connector-designer/allow-list.txt` file. You specify the allowed hosts in the file, one host per line. For example:

```
https://example.com
https://example2.com
```

2. An endpoint or database might require an TLS connection. You can specify the trust certificates that the connectors can use.

   The certificate must be in PEM format and the content of the certificate must be added to the `con-des-default/configuration/secrets/additional-trust-certificates.cer` file.

3. To configure the size of the Connector Designer container, for example the number of CPUs or available memory, edit the `analyze-deployment-tooling/runtime-environments/runtime-default.properties` file. For more information, see Configuring runtime environments.

4. To update Connector Designer with your changes, run:

```
./scripts/deploy
```

What to do next

To configure custom authentication methods for connectors to connect to external sources, see Configuring custom functions in Connector Designer.

If you need to change the values that you provided to the installation script, you can re-run the installation script. The script will prompt you to provide the values again.

Other commands that you can run are:

To stop Connector Designer, run:

```
./scripts/manage-environment -t stop
```

To restart Connector Designer, run:

```
./scripts/deploy
```

To repair or update Connector Designer, re-run the installation script:

```
./install-connector-designer.sh
```

Connector Designer deployment architecture

Connector Designer is a web application that allows users to create connectors. Connector Designer, and the connectors that it deploys, are deployed on a container separately from the i2 Analyze deployment. A load balancer, also running in a container, is used to route traffic to the i2 Analyze deployment and to Connector Designer.

When a connector is created and deployed using Connector Designer, the connector is run in a container and communicates with i2 Analyze by using the load balancer.

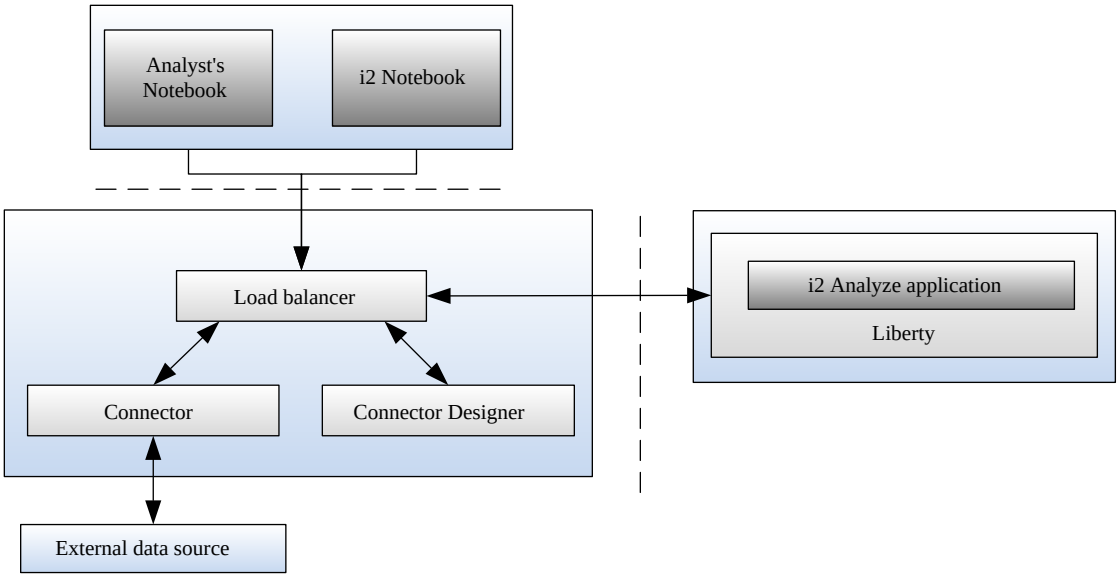
When you install Connector Designer, the required certificates and certificate stores are created. When a user deploys a connector from Connector Designer, it is deployed with the required certificates and certificate stores.

When a user deploys a connector from Connector Designer, the connector is deployed with the required certificates and certificate stores to communicate with i2 Analyze via the load balancer.

The following diagram shows the deployment architecture when Connector Designer is deployed alongside an existing i2 Analyze deployment.

Network

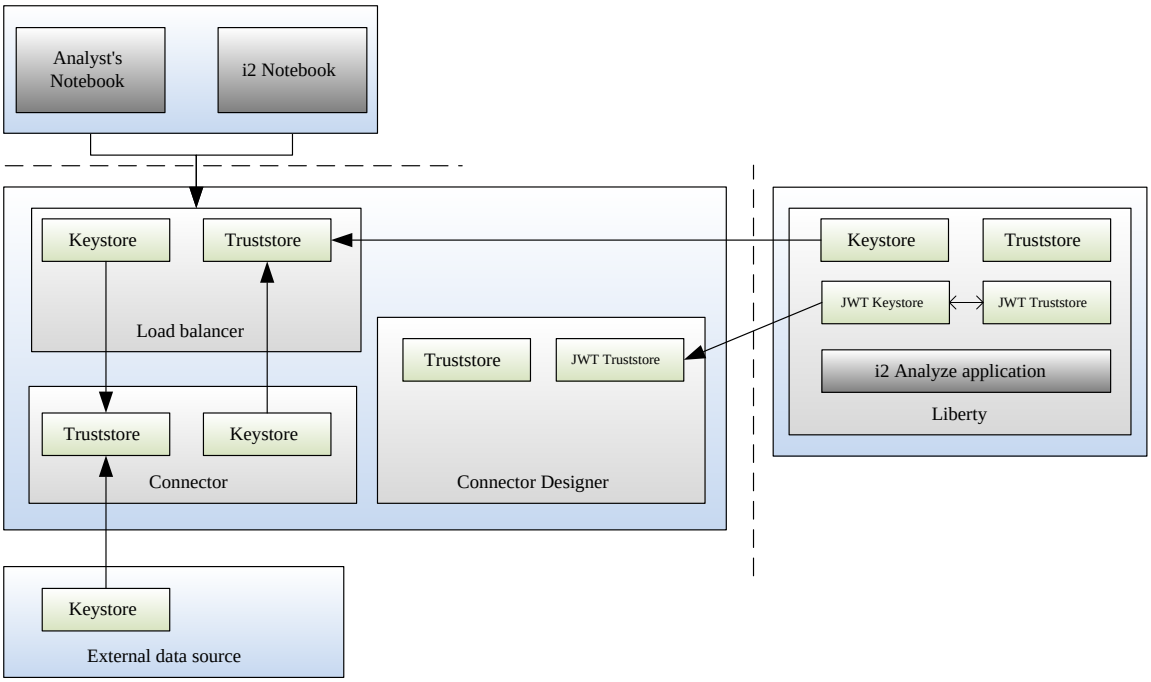
The following diagram shows the network architecture of Connector Designer:



When you deploy Connector Designer, you must provide the URLs of the Liberty servers in your i2 Analyze deployment. The load balancer uses these URLs to route traffic to the i2 Analyze deployment. If you already have a load balancer in your deployment, configure load balancer that is deployed as part of Connector Designer to point to your existing load balancer.

Security

The following diagram shows the security architecture of Connector Designer:



Connector Designer uses TLS to communicate and JWT tokens to authenticate users.

- For more information about configuring TLS, see Adding Connector Designer to your deployment
- For more information about configuring JWT tokens, see Configuring JWT authentication

Configuring custom functions in Connector Designer

When a user is creating a data source in Connector Designer, the user can select an authentication method from a list of predetermined options or dynamically added options. These dynamic options are provided by an administrator using *Custom functions*. Custom functions are used in situations where a static header cannot be set for a particular API call and a process is required to provide headers and their values at runtime.

The custom function is run before the API call is made, allowing the headers to be changed, removed, or added. After the headers are processed by the function, the API call continues with the resolved headers.

To provide Connector Designer with custom functions, use the `configs/<config-name>/configuration/connector-designer/custom_functions/configuration.json` to provide the name and location of the functions. For example:

```
{
    "<function-name>": {
        "functionFile": "<path-to-function-js-file>"
    }
}
```

Where:

- `<function-name>` is the name of the function that is displayed as an authentication option to the user.
- `<path-to-function-js-file>` is the path to the JavaScript (.js) file containing the function definition.

The structure of the JavaScript file that contains the function definition must have the following structure:

```
function buildCustomAuthorizationHeaders (headers) {
  // Your implementation to return the headers
  return headers; // The returned headers that are used in the API call
};

module.exports = buildCustomAuthorizationHeaders;
```

To update Connector Designer with your changes, run:

```
./scripts/deploy
```

Troubleshooting Connector Designer

This topic provides information about troubleshooting issues that you might encounter when using Connector Designer.

Incorrect external CA certificate

Error message:

```
Failed to reload gateway on https://i2.my-organization:9443
*   Trying 10.1.50.195:9443...
* Connected to i2.my-organization (10.1.50.195) port 9443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /tmp/tmp.RLRUGCtnjv
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
} [5 bytes data]
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
} [512 bytes data]
* TLSv1.2 (IN), TLS header, Certificate Status (22):
{ [5 bytes data]
* TLSv1.3 (IN), TLS handshake, Server hello (2):
{ [94 bytes data]
* TLSv1.2 (IN), TLS handshake, Certificate (11):
{ [1750 bytes data]
* TLSv1.2 (OUT), TLS header, Unknown (21):
} [5 bytes data]
* TLSv1.2 (OUT), TLS alert, unknown CA (560):
} [2 bytes data]
* SSL certificate problem: self-signed certificate in certificate chain
* Closing connection 0
```

Ensure that the CA certificate that you placed in `analyze-deployment-tooling/environment-secrets/generated-secrets/certificates/externalCA/CA.cer` is correct. You must provide the certificate that signed the Liberty server's certificates in your i2 Analyze deployment.

Incorrect password for adt-admin

Error message:

```
Failed to reload gateway on https://i2.my-organization:9443
*   Trying 10.1.50.195:9443...
* Connected to i2.my-organization (10.1.50.195) port 9443 (#0)
```

```
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /tmp/tmp.ejG1U1t9DV
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
} [5 bytes data]
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
} [512 bytes data]
* TLSv1.2 (IN), TLS header, Certificate Status (22):
{ [5 bytes data]
* TLSv1.3 (IN), TLS handshake, Server hello (2):
{ [94 bytes data]
* TLSv1.2 (IN), TLS handshake, Certificate (11):
{ [1750 bytes data]
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
{ [300 bytes data]
* TLSv1.2 (IN), TLS handshake, Server finished (14):
{ [4 bytes data]
* TLSv1.2 (OUT), TLS header, Certificate Status (22):
} [5 bytes data]
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
} [37 bytes data]
* TLSv1.2 (OUT), TLS header, Finished (20):
} [5 bytes data]
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
} [1 bytes data]
* TLSv1.2 (OUT), TLS header, Certificate Status (22):
} [5 bytes data]
# ERROR: Validation errors detected, please review the above message(s).
* TLSv1.2 (OUT), TLS handshake, Finished (20):
} [16 bytes data]
* TLSv1.2 (IN), TLS header, Finished (20):
{ [5 bytes data]
* TLSv1.2 (IN), TLS header, Certificate Status (22):
{ [5 bytes data]
* TLSv1.2 (IN), TLS handshake, Finished (20):
{ [16 bytes data]
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=i2.my-organization
*  start date: Jul 29 14:47:53 2024 GMT
*  expire date: Nov  1 14:47:53 2026 GMT
*  subjectAltName: host "i2.my-organization" matched cert's "i2.my-
organization"
*  issuer: CN=toolkit-ca
*  SSL certificate verify ok.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade:
 len=0
* TLSv1.2 (OUT), TLS header, Unknown (23):
} [5 bytes data]
* TLSv1.2 (OUT), TLS header, Unknown (23):
} [5 bytes data]
* TLSv1.2 (OUT), TLS header, Unknown (23):
} [5 bytes data]
* Server auth using Basic with user 'adt-admin'
* Using Stream ID: 1 (easy handle 0x555d46245810)
* TLSv1.2 (OUT), TLS header, Unknown (23):
} [5 bytes data]
> POST /opal/api/v1/gateway/reload HTTP/2
```

```
> Host: i2.my-organization:9443
> authorization: [REDACTED_PASSWORD]
> user-agent: curl/7.76.1
> accept: */*
> origin: https://i2.my-organization:9443
> content-type: application/json
>
* TLSv1.2 (IN), TLS header, Unknown (23):
{ [5 bytes data]
* TLSv1.2 (OUT), TLS header, Unknown (23):
} [5 bytes data]
* TLSv1.2 (IN), TLS header, Unknown (23):
{ [5 bytes data]
* TLSv1.2 (IN), TLS header, Unknown (23):
{ [5 bytes data]
< HTTP/2 401
< content-language: en
< content-length: 0
< date: Wed, 07 Aug 2024 10:51:33 GMT
<
{ [0 bytes data]
```

Ensure that the password for the `adt-admin` user in the Liberty server is correct. You can check the password in the `user.registry.xml` file matches the password in the `analyze-deployment-tooling/environment-secrets/generated-secrets/application/admin_PASSWORD` file.

Item type conversion

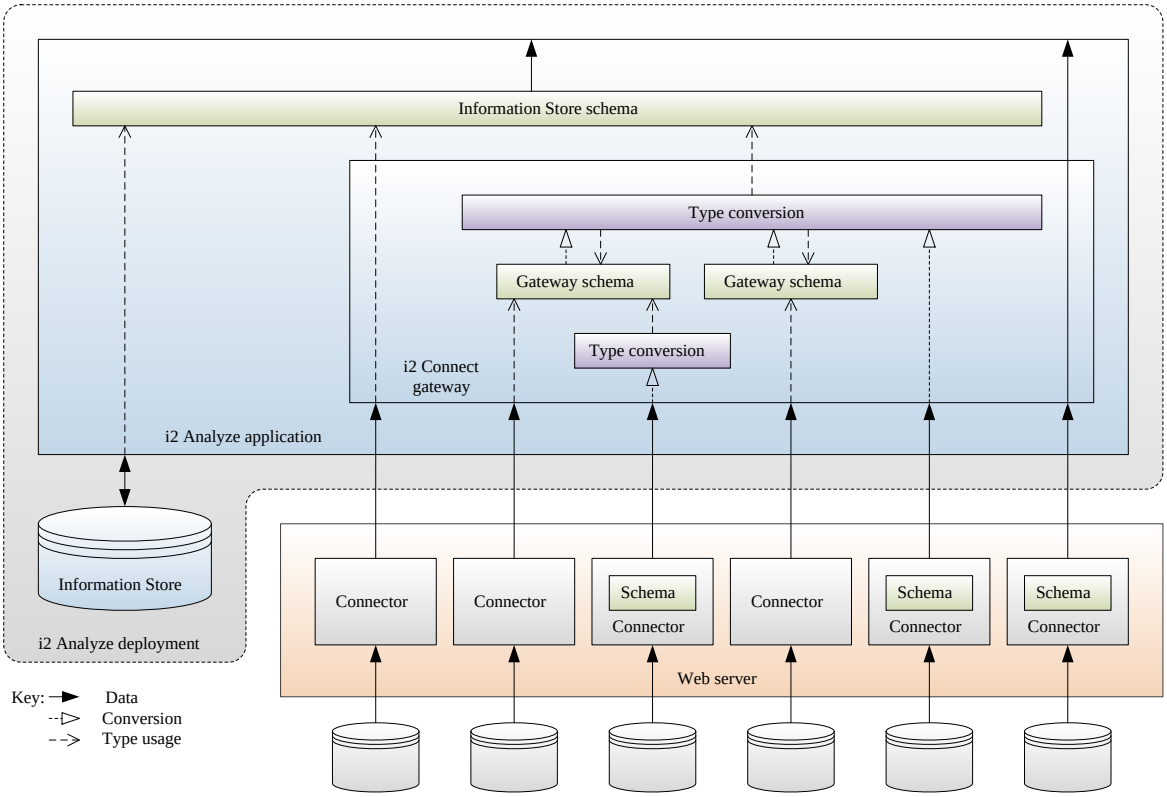
A deployment of i2 Analyze that includes the i2 Connect gateway can have multiple connector and gateway schemas, as well as an Information Store schema. In such a deployment, item types from different schemas can have similar names or purposes. i2 Analyze provides a way to convert between these types on the server, giving users a more consistent experience.

For example, two connector or gateway schemas might independently define "Person" item types, which might in turn be slightly different from a "Person" type defined in the Information Store schema. This kind of arrangement can have a number of consequences:

- The existence of several item types with the same or similar names can be confusing for administrators and for users, who see records on their charts and item types in their palettes that look the same, but are not.
- By the same token, the analytical tools in i2 Analyze and the Analyst's Notebook desktop client do not consider records with similar-looking types to be equivalent for the purposes of comparison and other functions.
- Only records that have types from the Information Store schema can be uploaded to the Information Store.

To address these problems, the i2 Connect gateway provides a dynamic type conversion service. At startup, the gateway builds a catalog of all the item types in all the schemas in the deployment. You can then create mappings from one item type to another, in the following combinations:

- Connector schema type to Information Store schema type
- Connector schema type to gateway schema type
- Gateway schema type to Information Store schema type
- Gateway schema type to gateway schema type (in a different gateway schema)



Type conversion mappings describe how the property values of source records that have a particular item type become the property values of target records that have a different item type.

For example, a gateway schema might define an entity type named "Person", with property types including "Forename" and "Surname". In the same deployment, the Information Store might also define an entity type named "Person", with property types including "First Name" and "Family Name". Provided that the property types in question have compatible logical types, you can create a mapping to change the type of any record with the gateway type so that it becomes a record with the Information Store type.

The effect of this type conversion mapping is that any record that would otherwise appear in search results or on the chart surface with the gateway type instead appears with the Information Store type. As a result, it can be compared directly with existing records in the Information Store, and can itself be uploaded to the Information Store if the user requires it.

Note: It is not mandatory to create mappings between item types. By default, records from connectors are copied to users' charts with their assigned entity or link types, without modification.

Creating type conversion mappings

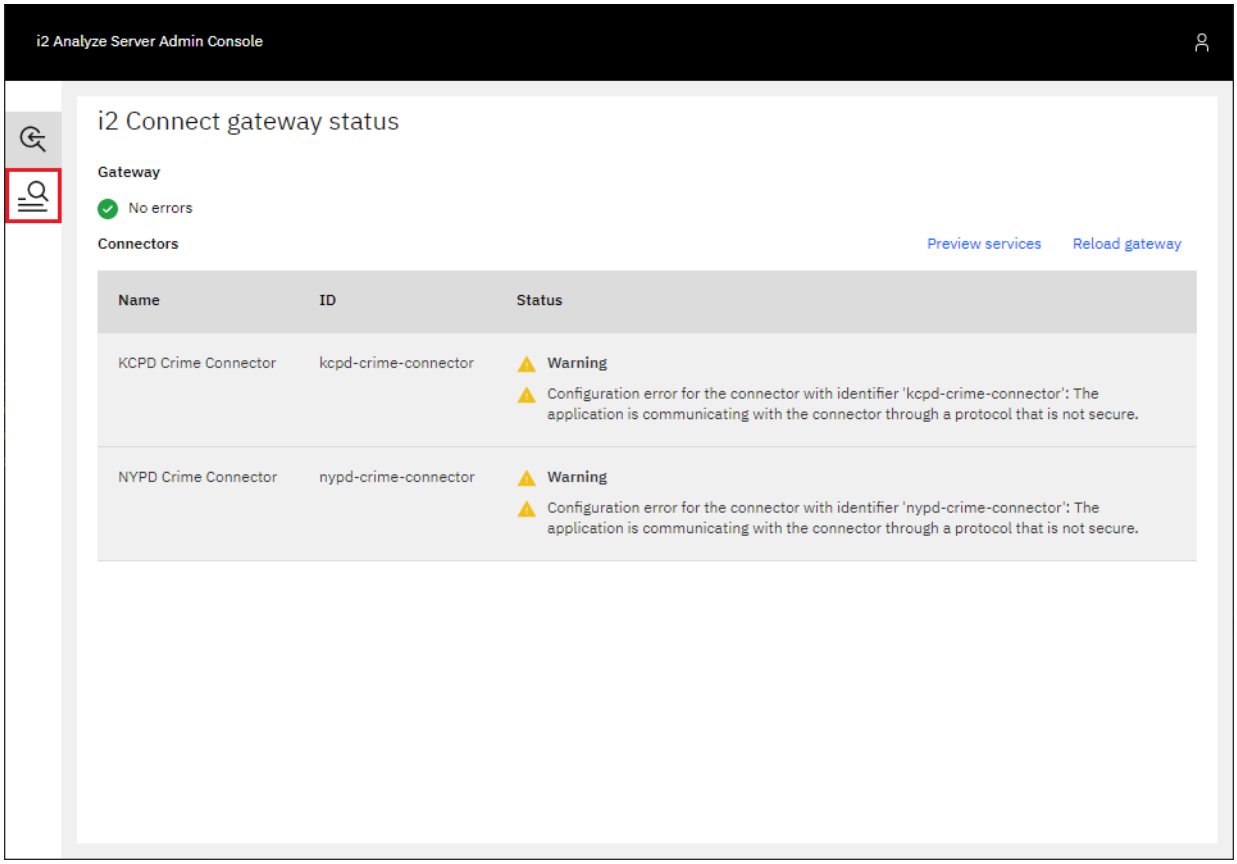
To create type conversion mappings, you use the i2 Analyze Server Admin Console.

In a web browser, navigate to the URI of a i2 Analyze development environment, but append `/admin`. For example: `http://<host_name>:9082/opal/admin`.

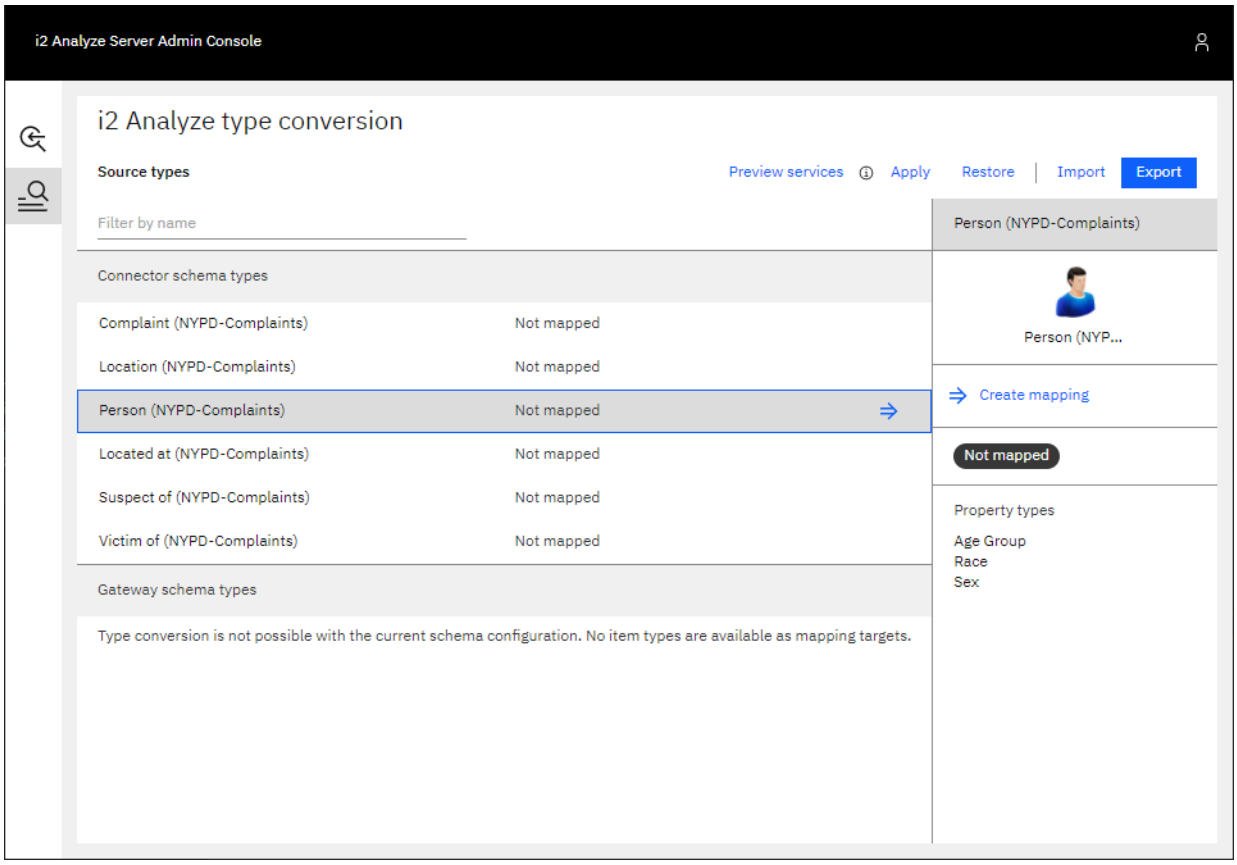
When you log in using credentials with the `i2:Administrator:Connectors` command access control permission, the i2 Analyze Server Admin Console appears. For more information about ensuring you have the correct permissions, see Enabling access to the Server Admin Console.

The mapping configuration interface

By default, the admin console displays the status of the i2 Connect gateway. To display the mapping configuration interface, select the **i2 Analyze Type Conversion** app, which is highlighted in red:



The console now displays a list of all the item types across all gateway and connector schemas, with some information about any item type mappings that have been defined. In this example, no mappings are configured yet.



Defining a mapping

Note: The following description is based on types from the `NYPD-Complaints` and `KCPD-Crime` schemas that feature in examples in the analyze-connect repository.

The `NYPD-Complaints` and `KCPD-Crime` schemas both contain entity types named "Person". Both types define the same property types: age, sex, and race. But a significant difference is that the NYPD Person type models age in ranges, while the KCPD Person type holds specific values for ages.

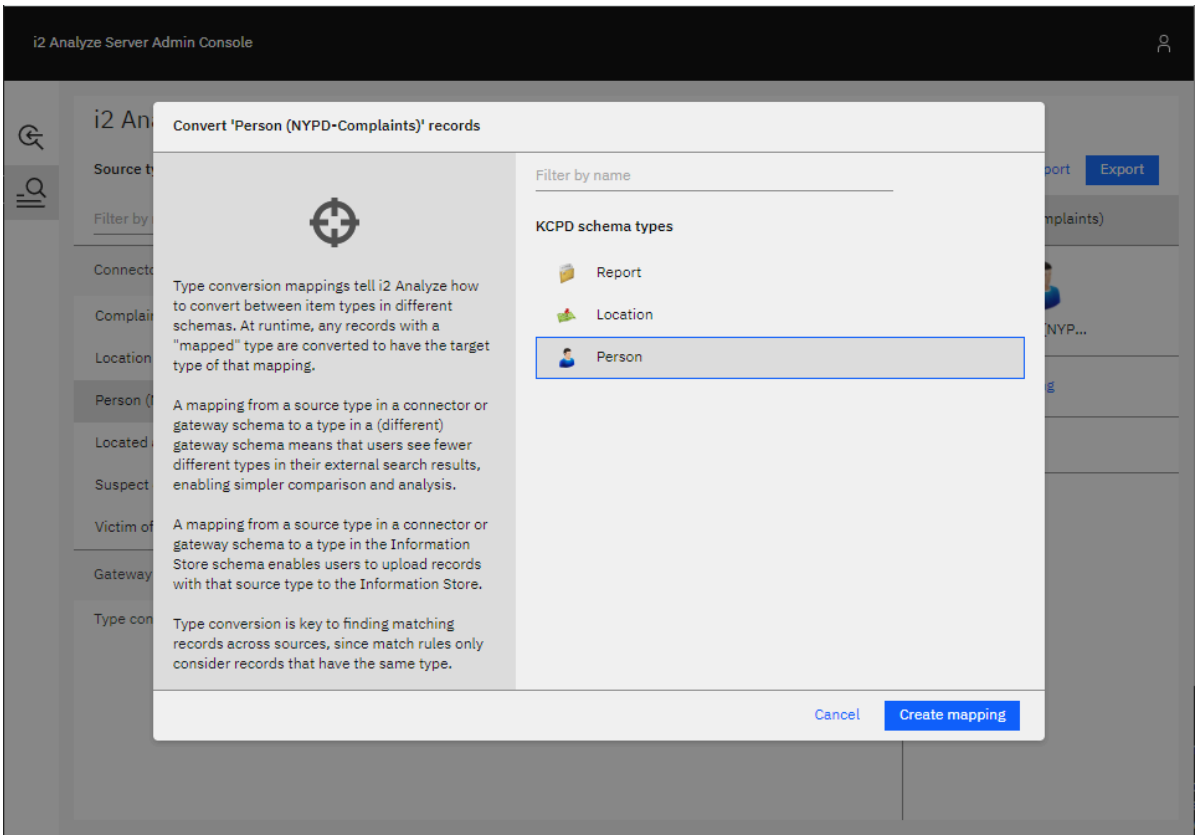
NYPD Person	KCPD Person
 Sex:F Age: 25-44 Race: Hispanic	 Race: Hispanic Sex: F Age: 29

1. Select the source type.

   In the list of item types, select the one to map from. For this example, the Person type from the `NYPD-Complaints` connector schema is selected. Then, click on either the arrow shown on the list item, or **Create mapping** in the pane on the right.

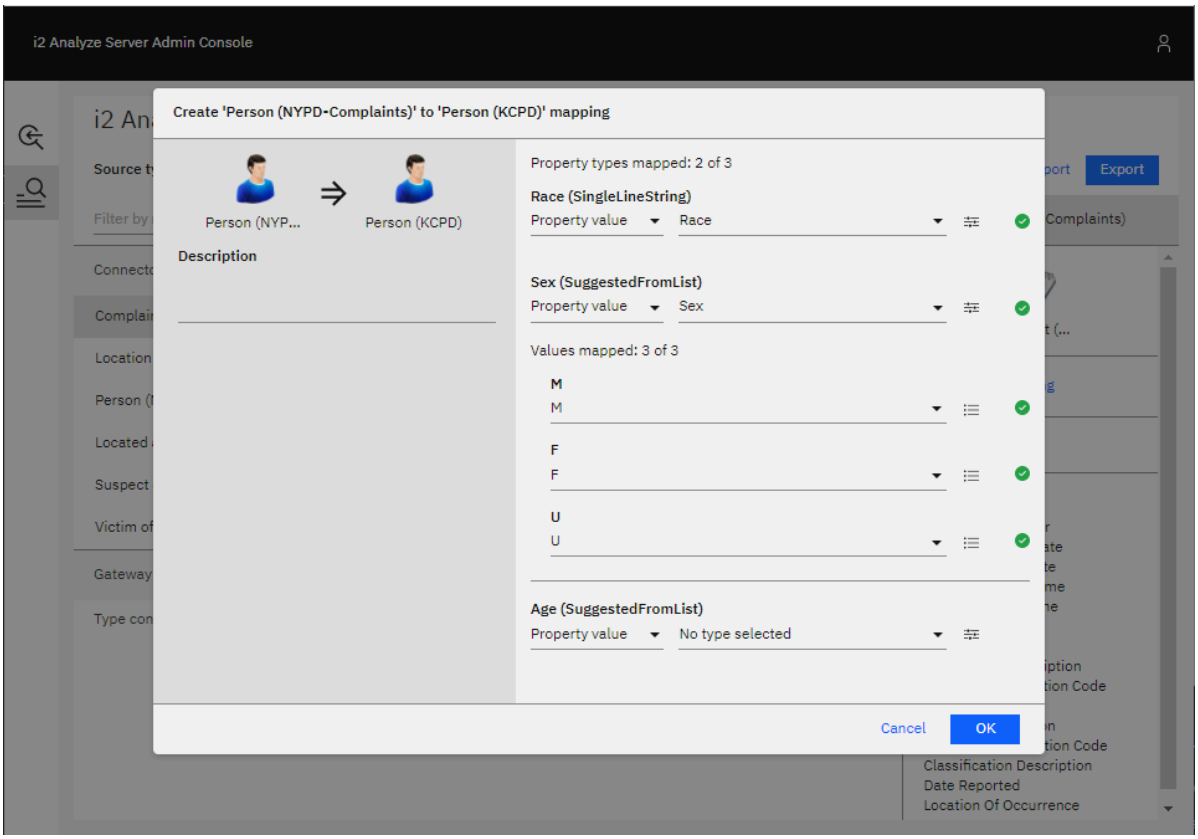
2. Select the target type.

   The app now displays a dialog box that contains a list of item types to which your selected source type can be mapped. Choose an appropriate item type. For this example, the Person type from the `KCPD-Complaints` gateway schema is selected. After you select the target type, click **Create mapping**.



3. Define the property mappings.

   At this stage, define how the property values of the source type are mapped to property values of the target type.



The properties shown in the dialog are the properties of the *target* type. For each target property, you can do one of the following:

- **Map a property from the source item type to the target property.** When a record of the source item type is converted to a record of the target item type, the target property is populated with the value of the source property. The logical type of the source property type must be compatible with the logical type of the target property type.
- **Map to a fixed value.** When a record of the source item type is converted to a record of the target item type, the target property is populated with the fixed value provided in the mapping configuration.
- **Leave the property unmapped.** When a record of the source item type is converted to a record of the target type, the target property is not populated. Target property types that are mandatory cannot be left unmapped.

Automatically mapped properties

A property type in the source item type is automatically mapped to a property type in the target item type if they have the same:

- Display name
- Logical type

If the two properties have the logical type SELECTED_FROM or SUGGESTED_FROM, any possible values of the source property that are also present in the target property are mapped automatically to those values in the target property.

You can see in the example that this logic was applied to the race and sex property types. These automatically generated mappings can still be edited if they do not meet your requirements.

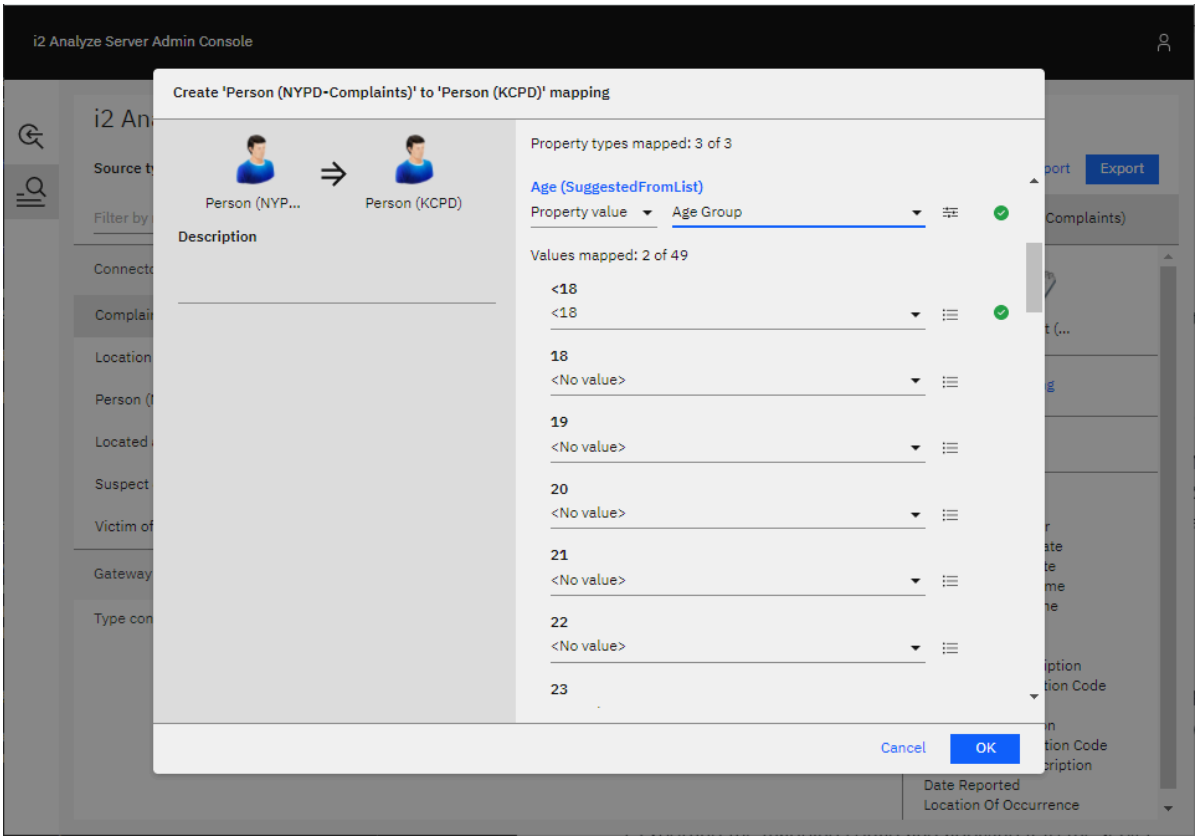
Mapping a property from the source item type

To map a property from the source item type to a target property:

a. Select **Property value** in the left-most dropdown menu below the target property name.

b. Choose a property from the source item type in the right-most dropdown. Only compatible properties from the source type are shown.

For all logical types except SELECTED_FROM and SUGGESTED_FROM, these are the only required steps. For properties with a logical type of SELECTED_FROM or SUGGESTED_FROM, you might also need to specify how the values are mapped.



Value mapping is necessary in this example, for the mapping of the age group property in the NYPD schema to the age property in the KCPD schema. The source age group property is a SUGGESTED_FROM list of age groups; the target age property is a SUGGESTED_FROM list of specific ages.

The values in the dialog box are the possible values for the target age property. For each target value, you can do one of the following:

- **Map a value from the source property to it.** Select which value of the source property is to be mapped to the target value. You cannot map a single value from the source property to more than one value of the target property.
- **Leave it unmapped.** Select **<No value>** in the dropdown. Then, no value from the source property type is mapped to this value of the target property.

Similar to how property types are automatically mapped, values of `SELECTED_FROM` and `SUGGESTED_FROM` properties are automatically mapped if the value of the target property type is the same as a value of the source property type. This is the case here for the "<18" and "65+" values.

For the other target values, you must either choose an age group from the source value to effectively collapse to a single age, or leave them unmapped.

Mapping a fixed value

To map a fixed value to a target property:

a. Select **Fixed value** in the left-most dropdown menu below the target property name.

b. Enter a fixed value to use in the left-most field.

Choosing a value for reverse conversions

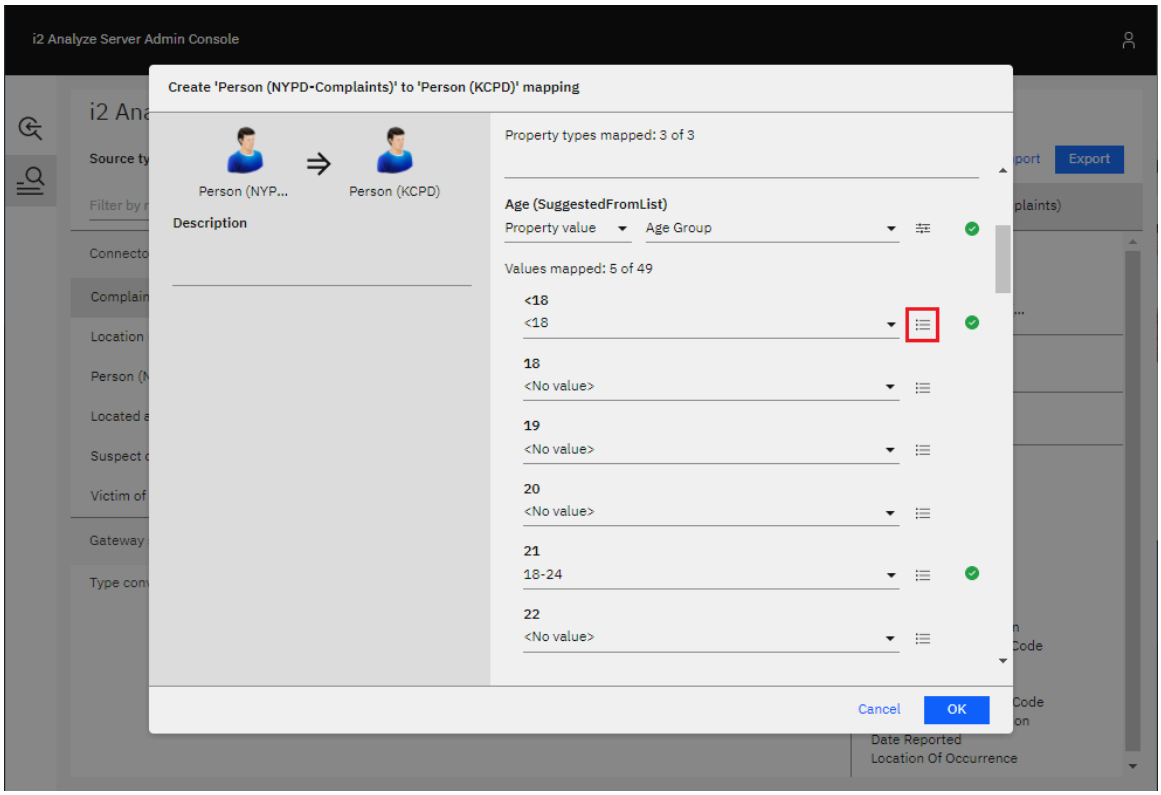
If a connector accepts a particular item type for the seeds of a seeded search, and that item type is the source type of a mapping, then users can use records of the mapping's target type as seeds when using the service.

When i2 Analyze sends these seed records to the connector, it must apply the mapping in reverse. The connector is expecting seeds of the source item type, and it is unaware of any mapping configured on the i2 Analyze server.

For mappings of properties where the logical type of both the source property and the target property are the same (and are not `SELECTED_FROM` or `SUGGESTED_FROM`), the reverse conversion just involves populating the source property with the value of the target property.

For mappings between `SELECTED_FROM` and `SUGGESTED_FROM` properties, you can choose how i2 Analyze maps the values of the target property to the possible values of the source property:

a. Click the button to the right of the dropdown menu for a target value that has been mapped



b. From the dropdown, select which one of the source values that you have mapped to the target value i2 Analyze should use when applying the mapping in reverse for seeded searches.

c. Confirm your choice by clicking **OK**.

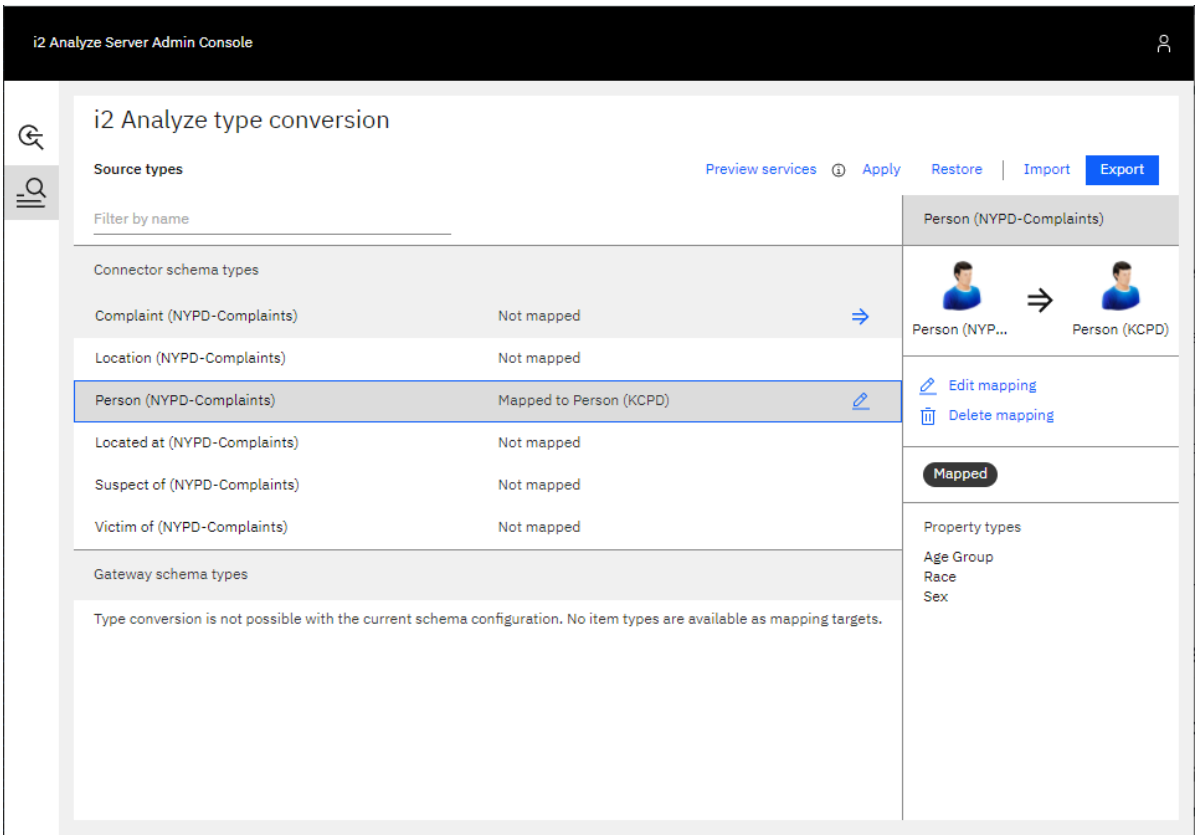
4. Add a description.

After you define the property mappings, you can add a description of the item type mapping.

5. Confirm the mapping.

Click **OK** in the bottom-right of the dialog to save your item type mapping. The list of item types shows how the source type has been mapped.

Note: At this point your mapping is not deployed on the server.

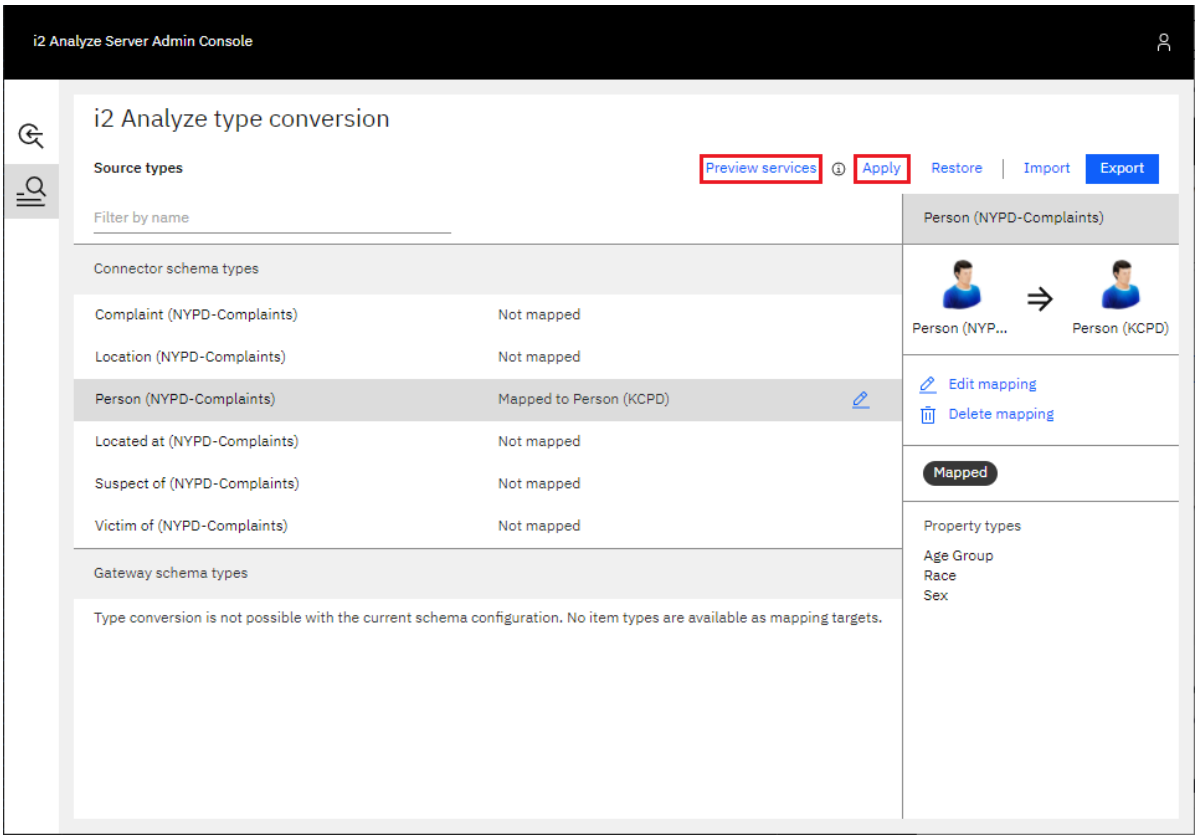


You can edit the mapping by clicking **Edit mapping** in the right-hand pane, or delete it by clicking **Delete mapping**.

Applying the mappings for testing

After you define the item type mappings, you can test the i2 Connect services in your i2 Analyze deployment to see how they would look if you were to apply the mappings you have configured. To do this:

1. Click **Apply**. This applies the current mapping configuration for testing.
2. Click **Preview services**. This opens the **External Searches** dialog as it would appear in Analyst's Notebook. You can use the services and see the results as they would appear if the mapping that has been applied for testing was deployed on the server.



To revert the mapping configuration back to the current deployed configuration on the server, you can also use the **Restore** button that is next to **Apply**.

Important: Property mappings can affect the behavior of i2 Connect services that support seeds with semantic constraints. Try to ensure that the target property types in your mappings have the same semantic types as (or child semantic types of) the source property types.

Applying the mapping to the server

After you preview the i2 Connect services using your new mappings and you are happy with them, you can deploy your mapping configuration to the i2 Analyze server. To do this:

1. In the mapping configuration interface, click **Export** in the top right corner. This downloads a `mapping-configuration.json` file.
2. Move the `mapping-configuration.json` file into the `toolkit/configuration/fragments/common/WEB-INF/classes` directory of your configuration.
3. Redeploy the i2 Analyze server:

```
setup -t stopLiberty
setup -t deployLiberty
setup -t startLiberty
```

Adding a connector with a connector schema to an existing deployment

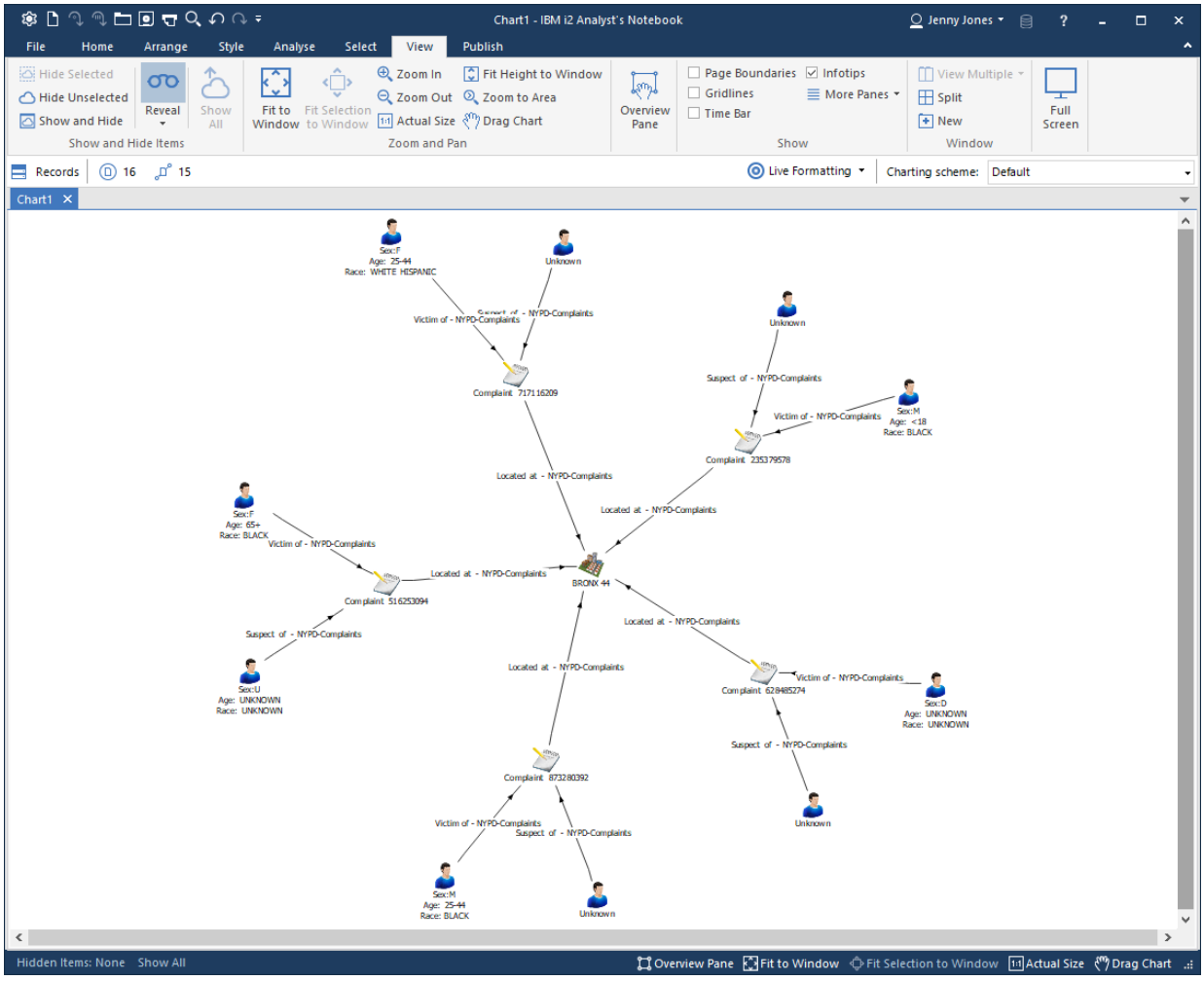
In this example scenario, we will add a new connector with its own connector schema to an i2 Analyze deployment with an existing gateway schema. Item types in the connector schema will be mapped to item types in the gateway schema where possible.

Note: These instructions can also apply to mapping item types from one gateway schema to another gateway schema.

Setting up the scenario

To follow this example, start by deploying i2 Analyze with the example NYPD connector, as described in the analyze-connect repository, with the connector configured to use a gateway schema.

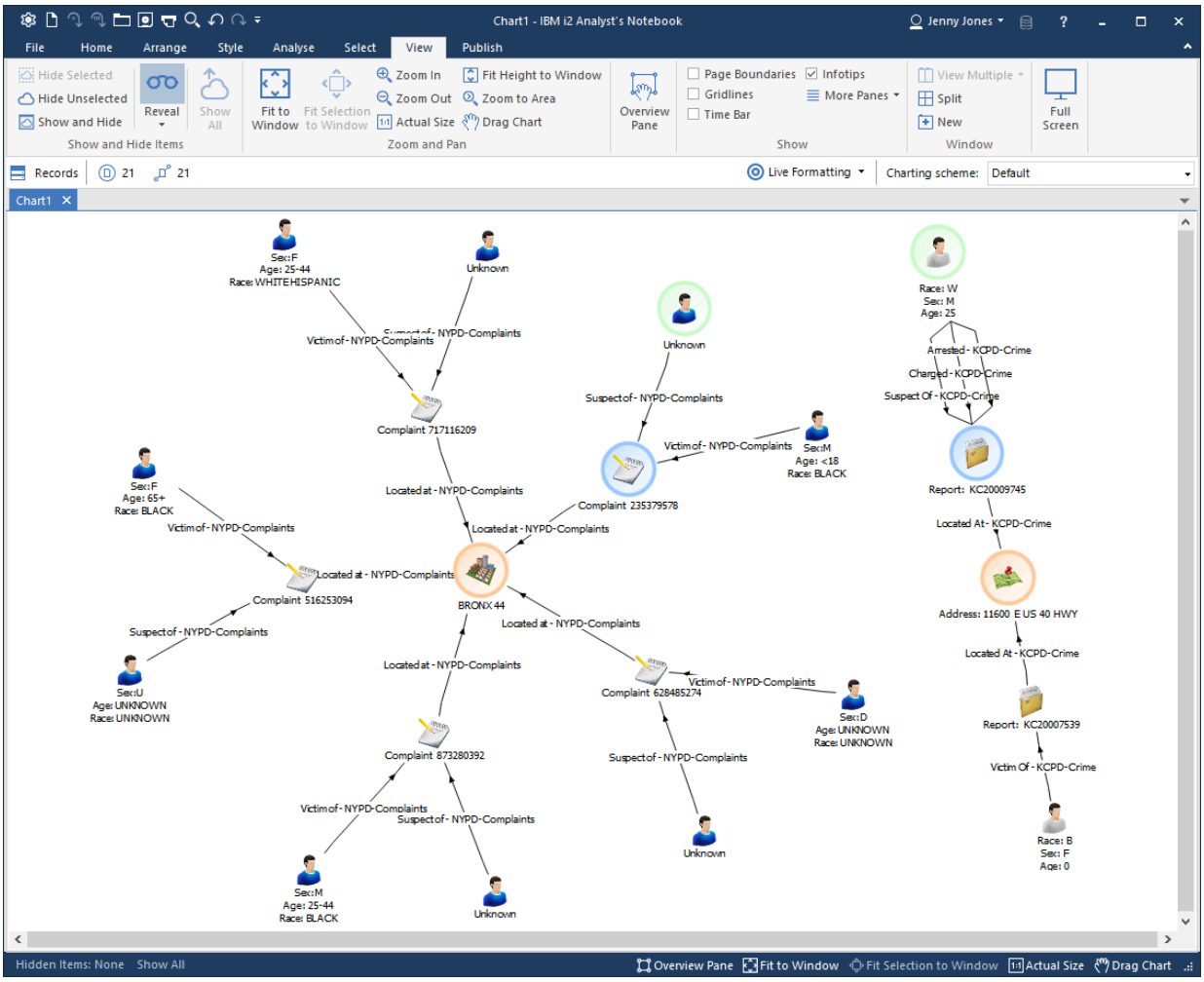
With the example deployment in place, you can connect to the i2 Analyze server from Analyst's Notebook and use the services provided by the NYPD connector. For example, you might use the "Get All" service and copy some of the results to a chart.



Adding a second connector

Next, add the example KCPD connector to the deployment, ensuring that it is configured to use its own connector schema.

When the connector is deployed, you should be able to use the services that it provides and copy some results to the chart.



The image uses highlighting and different icons to show how the gateway schema used by the NYPD connector and the connector schema used by the KCPD connector use different item types to model the same real-world objects.

Looking at all the available item types, you can see that both schemas have their own:

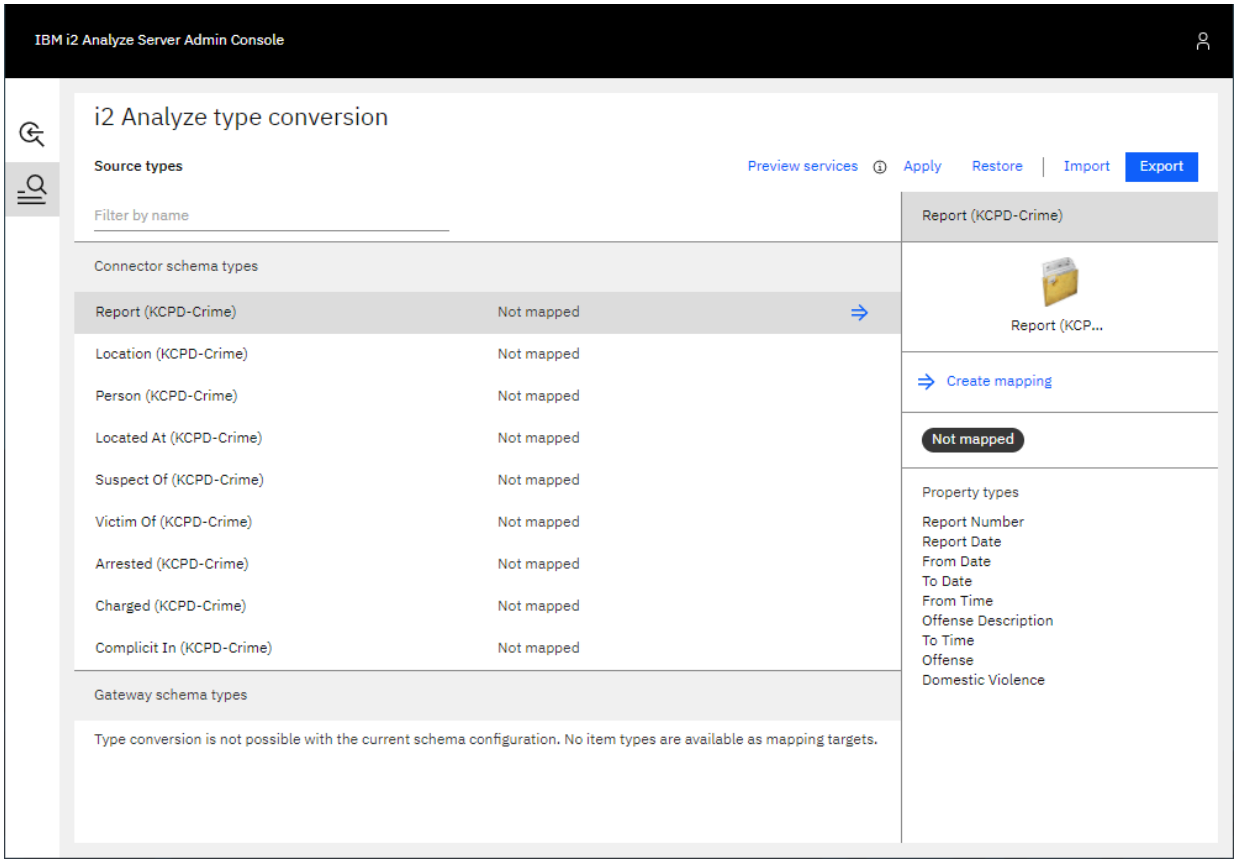
- 'Person' entity type
- 'Location' entity type
- 'Complaint'/'Report' entity type (different names, but modeling the same concept)
- 'Located At' link type

- 'Suspect Of' link type
- 'Victim Of' link type

You can use type conversion mappings to resolve this duplication.

Configuring item type mappings

In a web browser, go to the the **i2 Analyze Type Conversion** app in the i2 Analyze Server Admin Console to see the list of item types that can be mapped. You should see a list of all item types in the KCPD-Crime schema, and that none of them has been mapped.



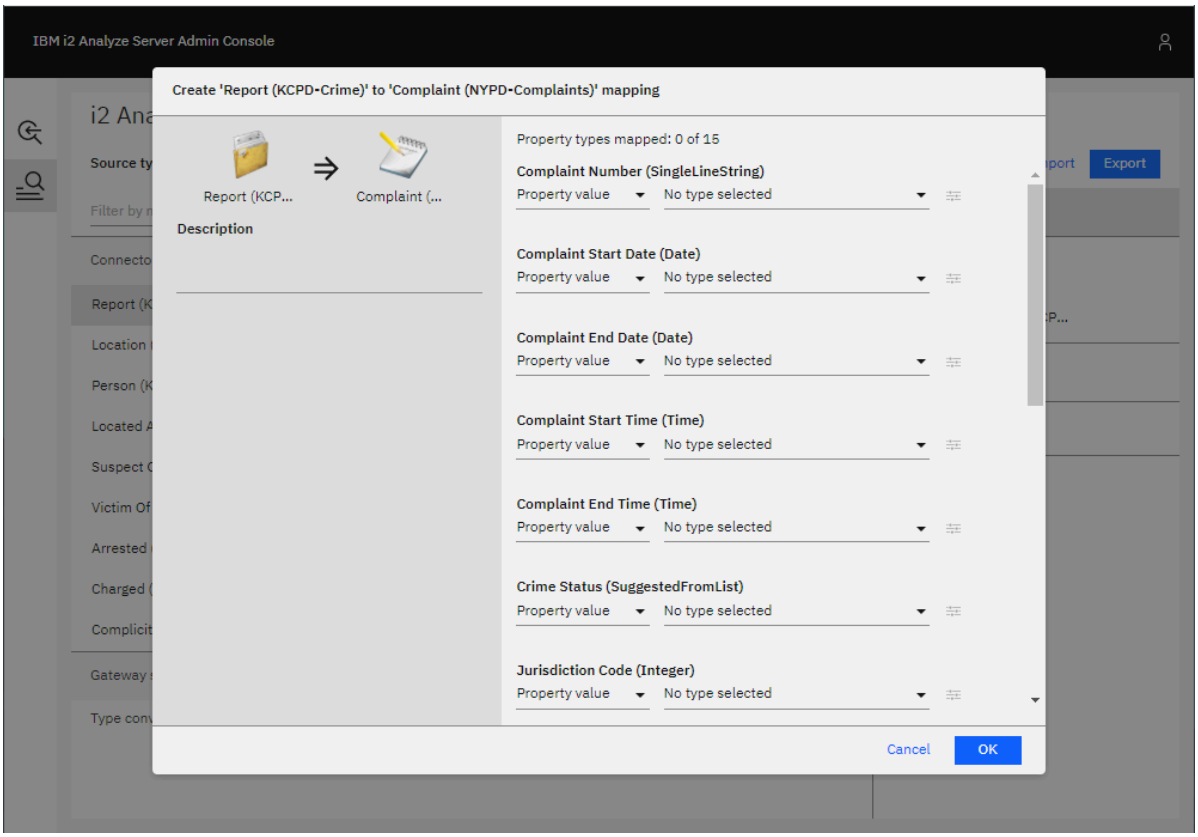
Where appropriate, KCPD-Crime item types can be mapped to similar item types in the NYPD-Complaints gateway schema.

Report and Complaint

The Report entity type in the KCPD-Crime schema can be mapped to the Complaint entity type in the NYPD-Complaints schema.

1. Select the **Report (KCPD-Crime)** item type in the list, then click **Create mapping** in the right-hand pane. This shows the list of item types to which the Report item type can be mapped.
2. Select the **Complaint** item type from the NYPD-Complaints schema, then click **Create mapping**. You will see all the property types of the Complaint type in the NYPD-Complaints schema, with

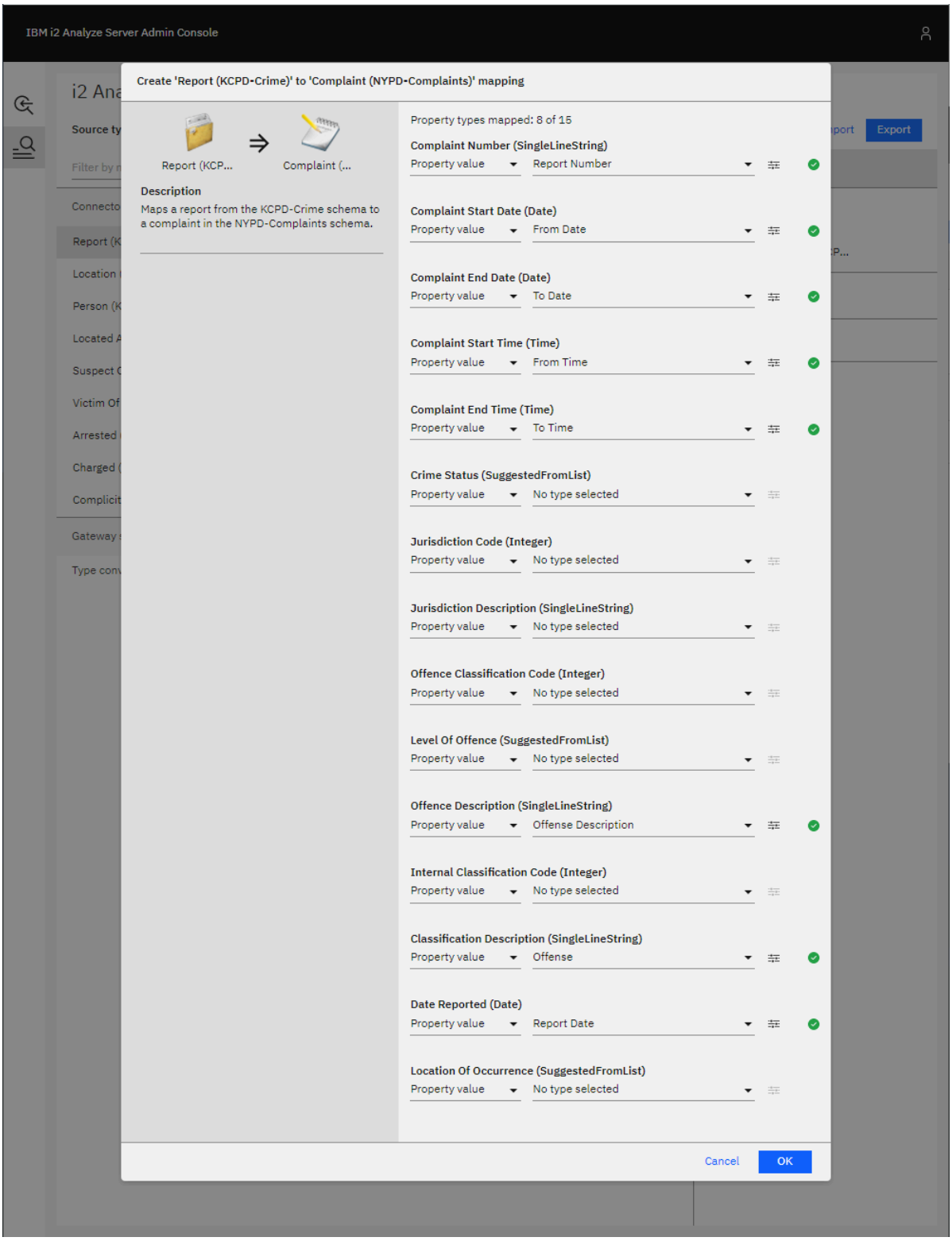
options available to choose how they should be populated when mapping from a KCPD-Crime Report record.



3. In the example below, the property types of the Report (KCPD-Crime) type are mapped to the property types of the Complaint (NYPD-Complaints) according to the following table.

Report (KCPD-Crime)	**Complaint (NYPD-Complaints)**
Report Number	Complaint Number
From Date	Complaint Start Date
To Date	Complaint End Date
From Time	Complaint Start Time
To Time	Complaint End Time
Offense Description	Offence Description
Offense	Classification Description

Report (KCPD-Crime)	**Complaint (NYPD-Complaints)**
Report Date	Date Reported

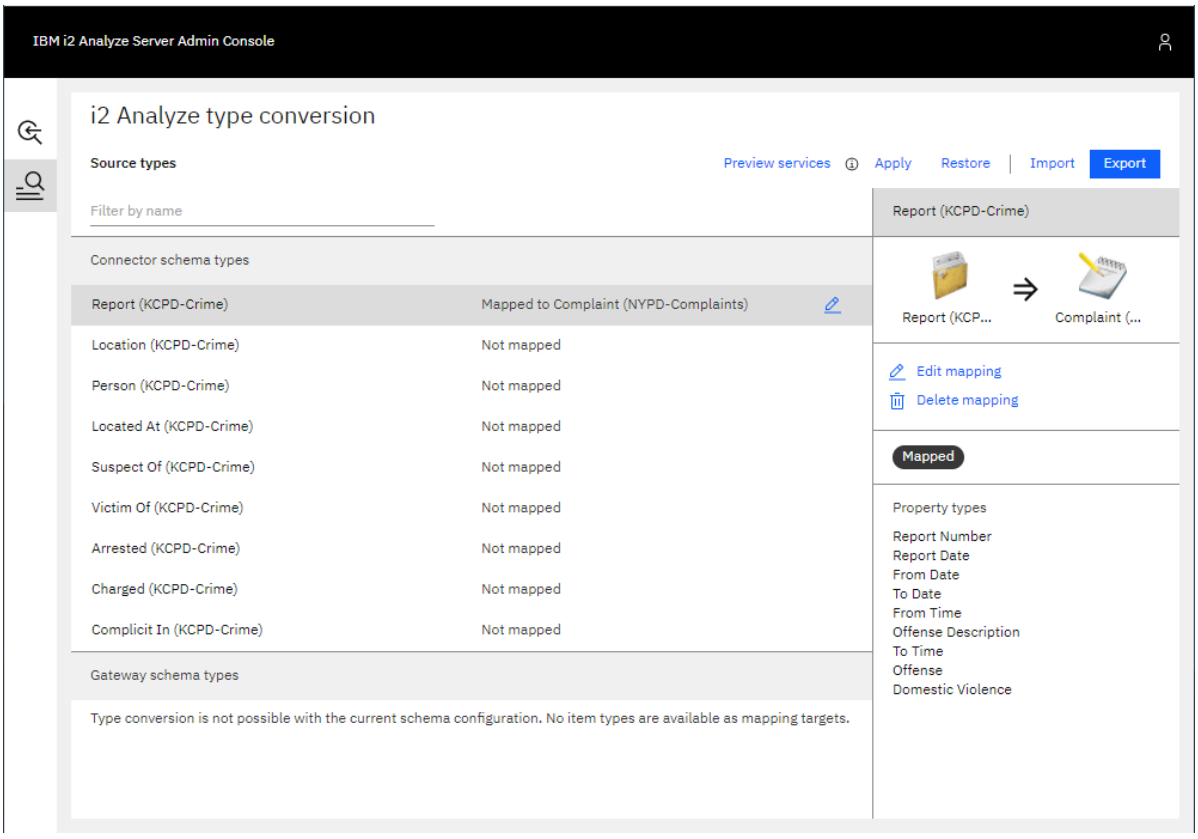


There is no comparable property type in the Report (KCPD-Crime) type for the following Complaint (NYPD-Complaints) property types:

- Crime Status
- Jurisdiction Code
- Jurisdiction Description
- Offence Classification Code
- Level Of Offence
- Internal Classification Code
- Location Of Occurrence

As a result, these types are left unmapped. The properties of Complaints records that are converted from Report records will not be populated.

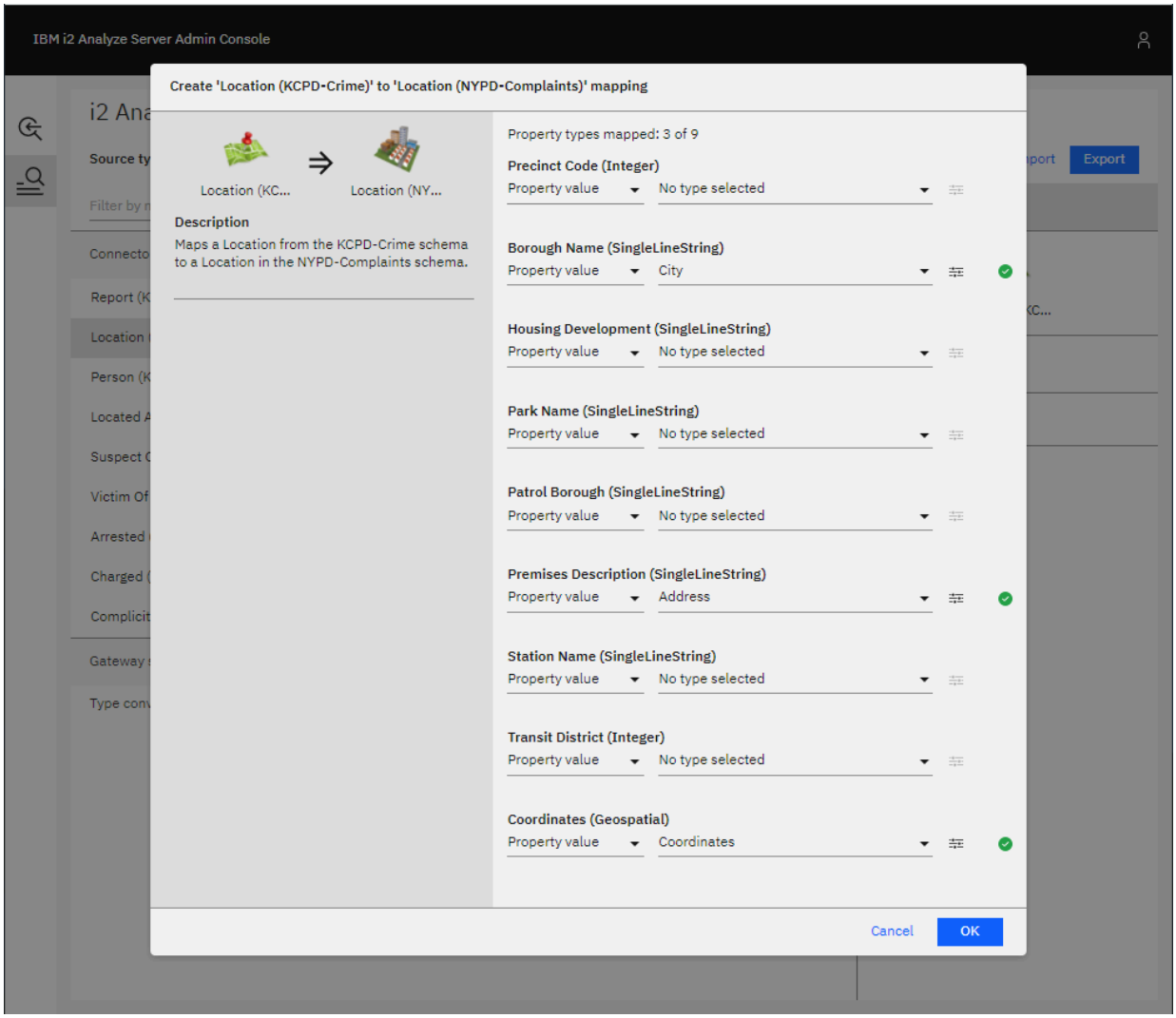
4. When you are satisfied with all the property type mappings, confirm the mapping by clicking **OK**. You should see in the list of types that Report (KCPD-Crime) has been mapped to Complaint (NYPD-Complaints).



Location

Following the same process, the Location type from the KCPD-Crime schema can be mapped to the Location type in the NYPD-Complaints schema.

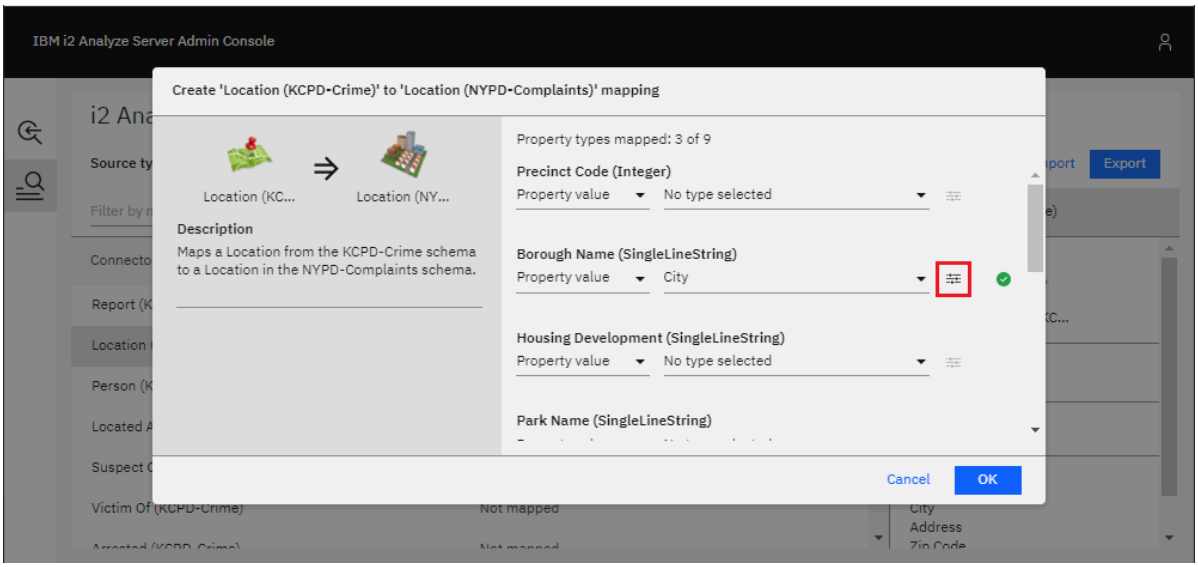
The Coordinates property type of Location (KCPD-Crime) is automatically mapped to the Coordinates property type of Location (NYPD-Complaints). The image shows an example configuration that you could use.



You can see that the Borough Name properties of NYPD-Complaints Locations will be populated from the value of the City properties of KCPD-Crime Location records. These property types are not an exact match for one another, but this mapping at least makes it clear when Locations are in Kansas City rather than New York City, in case Coordinates are not provided.

You can go further and ensure the Borough Name property is always populated in converted Locations by setting a default value that will be used if the source KCPD-Crime Location record does not contain a value for the City property. To do this:

1. Click the button highlighted to the right of the Borough Name configuration shown below.



2. Enter a default value to use for the Borough Name property if a source KCPD-Crime Location record has no City value. For example, "Kansas City".
3. Click **OK**.

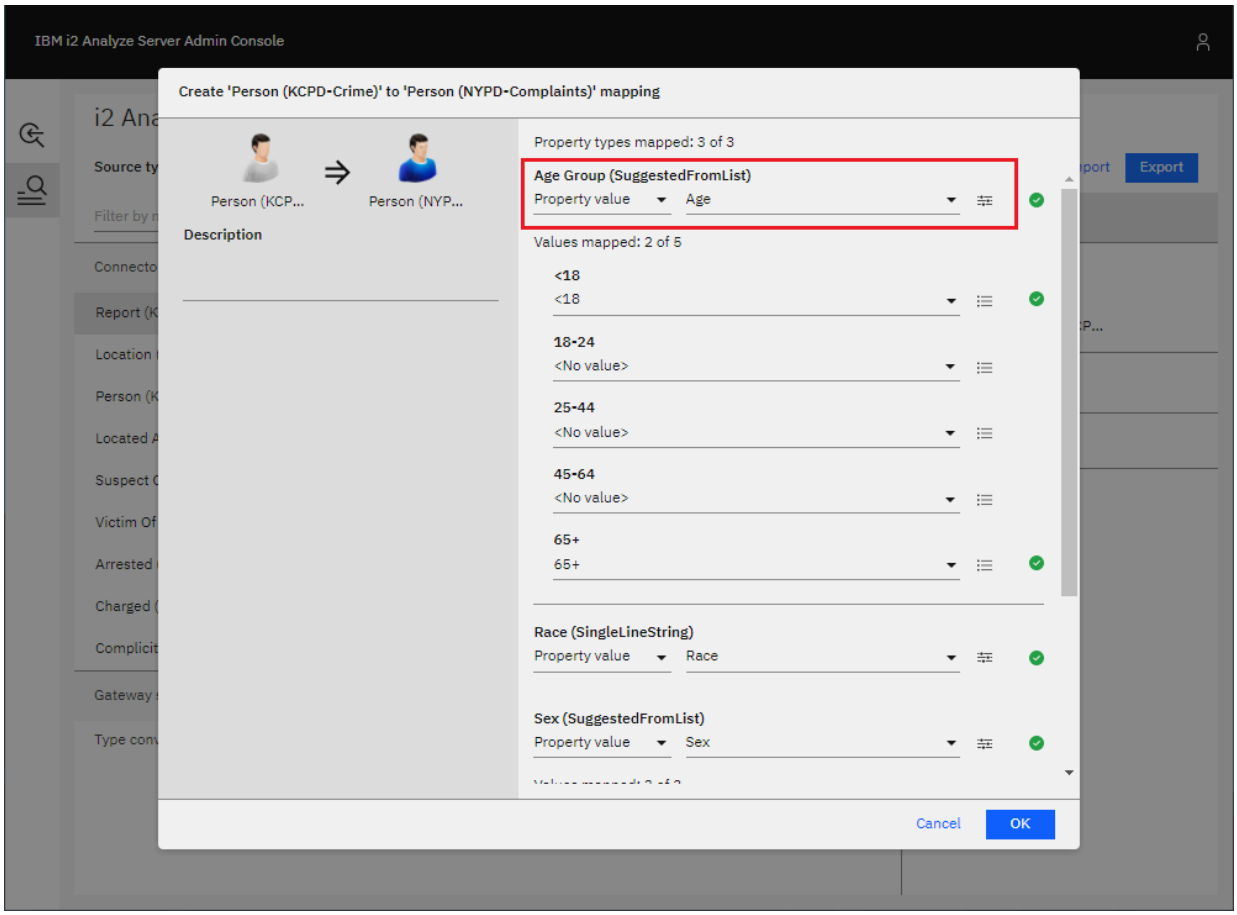
The mapping of Address to Premises Description does not seem like a perfect match either. But the Address of a Location might be too important not to have, and the Premises Description is a suitable target property in which to store it.

Again, when you are happy with the mapping configuration for Location records, click **OK** to confirm the mapping.

Person

You can follow the same process to create a mapping from the Person (KCPD-Crime) type to the Person (NYPD-Complaints) type.

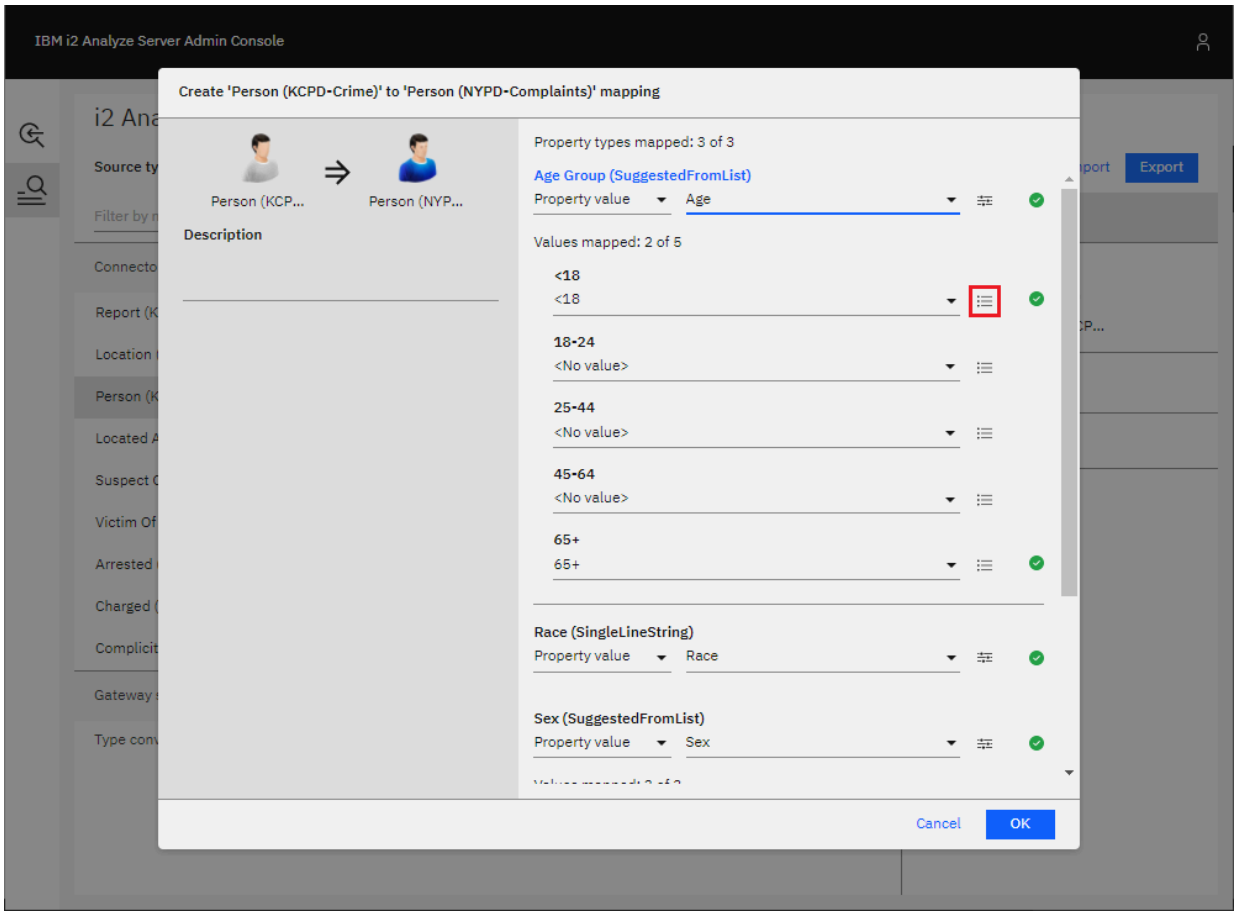
The Race and Sex property type mappings are generated automatically, so the only property type left is Age Group. You can map the values of the Age property type from the Person (KCPD-Crime) type, as shown below.



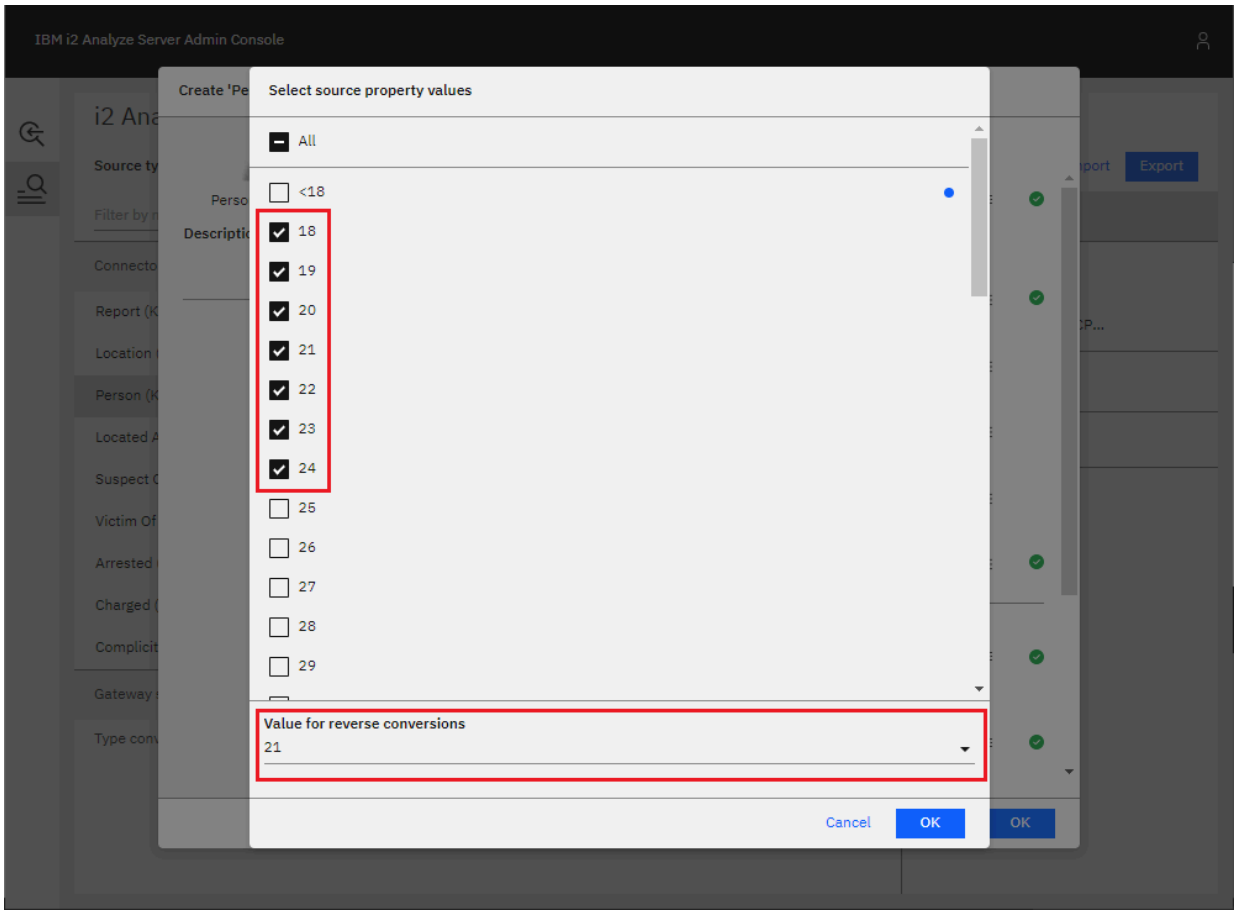
The Age property types of the source type and the target type are both `SUGGESTED_FROM` types, but they have different suggested values. The source property type, from the KCPD-Crime schema, has suggested values: `<18`, `19`, `20`, `21`, ..., `64`, and `65+`; whereas the target property type, from the NYPD-Complaints schema, has suggested values: `<18`, `18-24`, `25-44`, `45-64`, and `65+`.

The `<18` and `65+` values are automatically mapped to one another. For the remaining target values (`18-24`, `25-44`, and `45-64`), you must choose which source values should map to them.

You have the option to select a single value of the source property type from the dropdown, but that wouldn't make much sense. The source property values `18`, `19`, `20`, `21`, `22`, `23`, and `24` should *all* map to the `18-24` target value. To do this, click the button to the right of the dropdown menu beneath *18-24*, highlighted below.



Then, select all the appropriate values as shown below, and select a source value to use when the conversion is applied in reverse during any seeded searches.



Located At

The Located At (KCPD-Crime) link type can be mapped to the Located At (NYPD-Complaints) link type by following the same process:

1. Select **Located At (KCPD-Crime)** in the list of types, then click **Create mapping** in the right-hand pane.
2. Select **Located At** from the NYPD-Complaints link types, then click **Create mapping**.
3. There are no property types to map, so just add a description if you wish.
4. Confirm the mapping by clicking **OK**.

Suspect Of

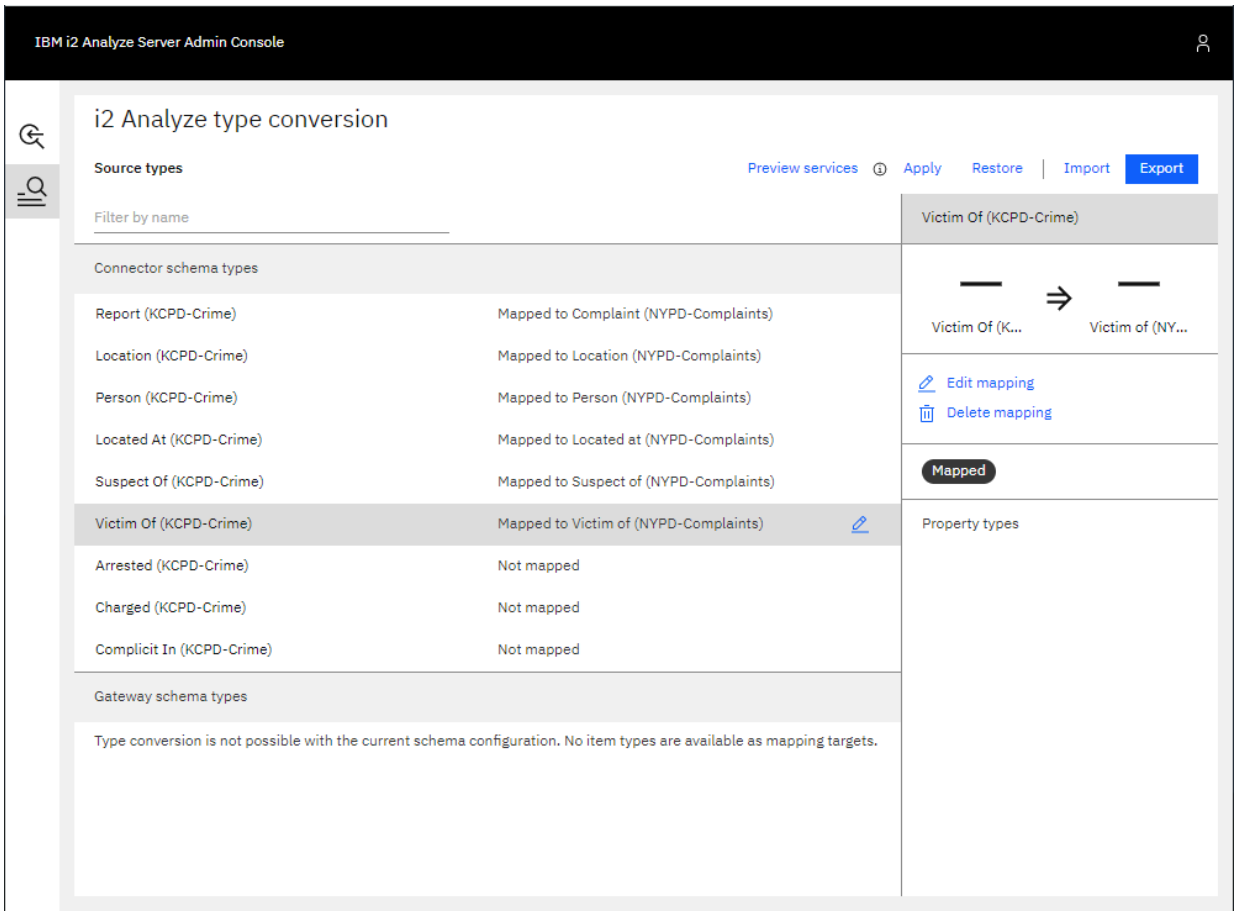
Map Suspect Of (KCPD-Crime) to Suspect Of (NYPD-Complaints) in the same way as Located At.

Victim Of

Map Victim Of (KCPD-Crime) to Victim Of (NYPD-Complaints) in the same way as Located At.

Testing the item type mappings

When you have defined the item type mappings, you should see an updated list of types like the example shown below.



The Arrested, Charged, and Complicit In link types are left unmapped, because there are no suitable target link types in the existing NYPD-Complaints schema. i2 Analyze will recognize that their end types have been mapped, and allow the NYPD-Complaints schema's Person and Complaint types to be connected by these links.

To test the item type mappings by previewing the external searches provided by the NYPD and KCPD connectors:

1. Click **Apply** in the top-right. This applies the mappings to the test environment that is available through the Admin Console. (It does not apply the mappings to the live server.)
2. Click **Preview services** to open a preview of how the services would behave with the mappings you have configured. Notice how the KCPD Connector's Get All service now returns item types from the NYPD-Complaints schema.
3. Go back and make any changes to the mappings, repeating steps 1 and 2 until you are satisfied with the configuration.

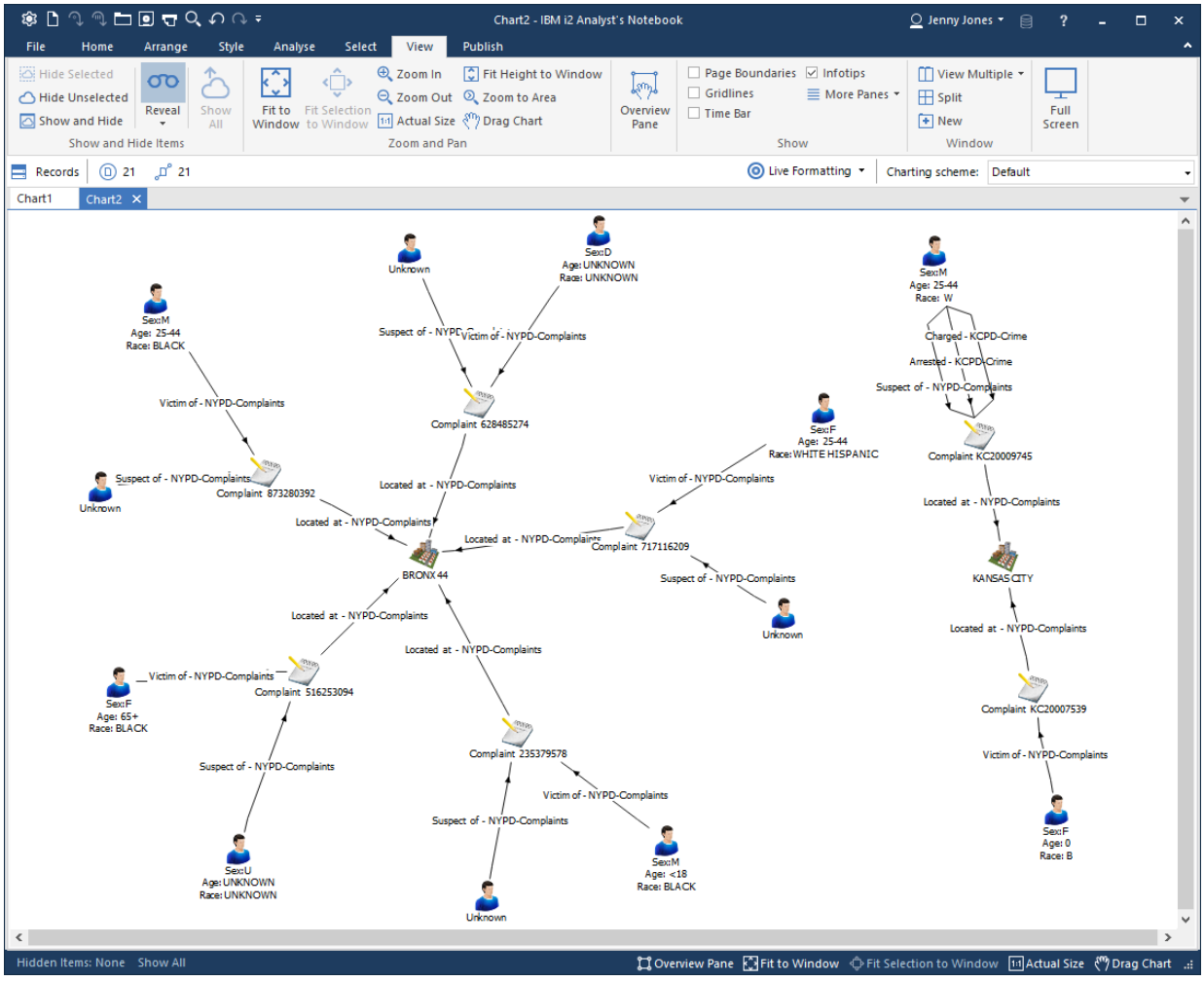
Applying the item type mappings to the i2 Analyze server

To apply the mapping configuration you have created on the i2 Analyze server for all users, see Applying the mapping configuration to the i2 Analyze server.

The result

In Analyst's Notebook, use some of the services provided by the two connectors and copy some results from each connector to your chart. Notice how there are now no duplicate item types.

For example, compare the chart below to the one at the beginning of this walkthrough. You can see that all of the records have types from the NYPD-Complaints schema, except for the unmapped Arrested and Charged links in the top-right corner.



Adding a connector with a connector schema to an existing deployment with a gateway schema and an Information Store schema

This example scenario demonstrates how to add a new connector with its own connector schema to an i2 Analyze deployment with an existing gateway schema and an Information Store schema. Item types in the connector schema are mapped to item types in the gateway schema, and item types in the gateway schema are mapped to item types in the Information Store schema.

The example also demonstrates how i2 Analyze resolves item type mappings that are chained together. That is, how mapping type *A* to type *B* and type *B* to type *C* results in an implicit mapping from type *A* to

type *C*, so that all records of type *A* are converted to records of type *C*. It further shows how mapping to item types in the Information Store schema allows converted records to be uploaded to the Information Store.

Setting up the scenario

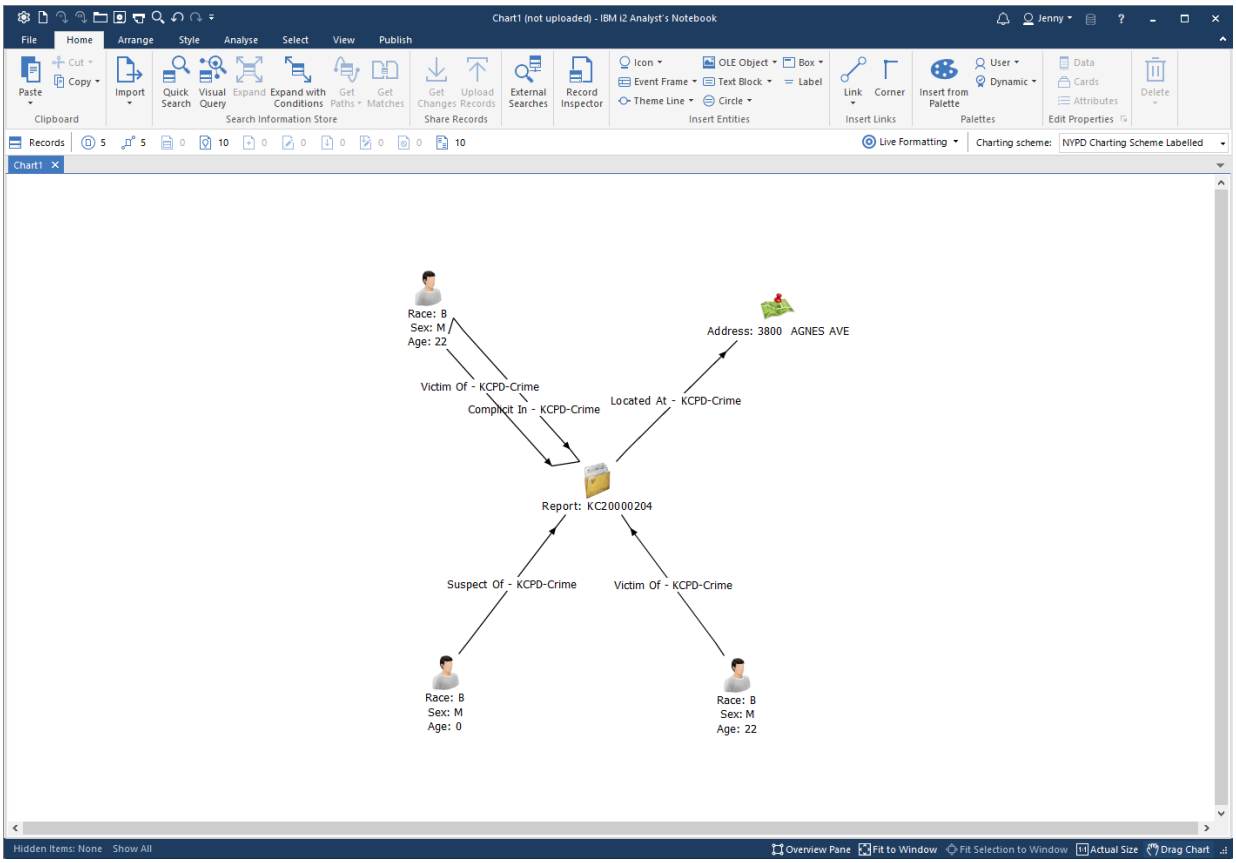
To follow this example:

1. Deploy i2 Analyze with the Information Store and the example NYPD connector, as described in the `analyze-connect` repository.

   Use the NYPD schema as the Information Store schema, and configure the connector to use it.

2. Add the example KCPD connector to the deployment, but configure this connector to use a gateway schema.

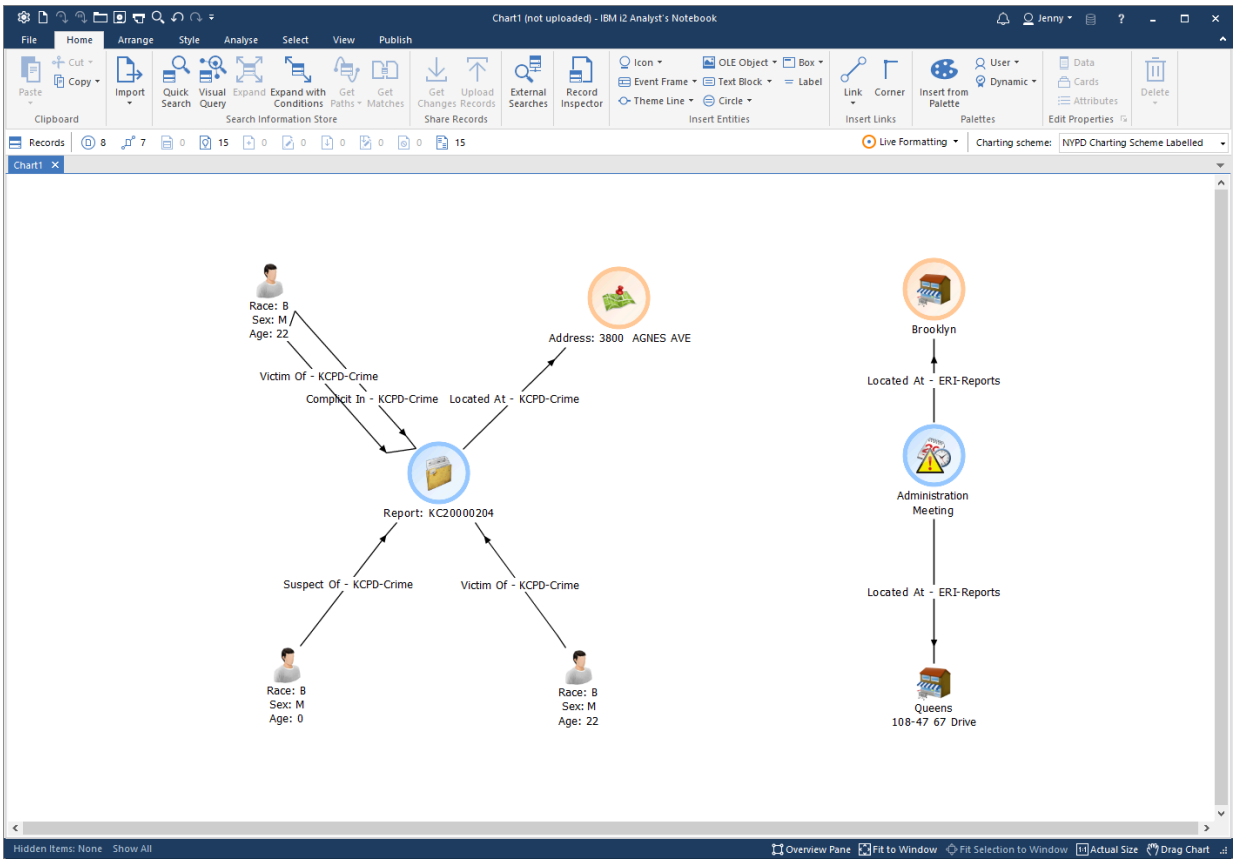
With the example deployment in place, you can connect to the i2 Analyze server from Analyst's Notebook and use the services provided by the KCPD connector. For example, you might use the "Get All" service and copy some of the results to a chart.



Adding a new connector with a connector schema

Next, add the example ERI connector to the deployment, ensuring that it is configured to use its own connector schema.

You can now use the services provided by the ERI connector, and copy results from those services to the chart too.



The image uses highlighting and different icons to show how the gateway schema used by the KCPD connector and the connector schema used by the ERI connector use different item types to model the same real-world objects.

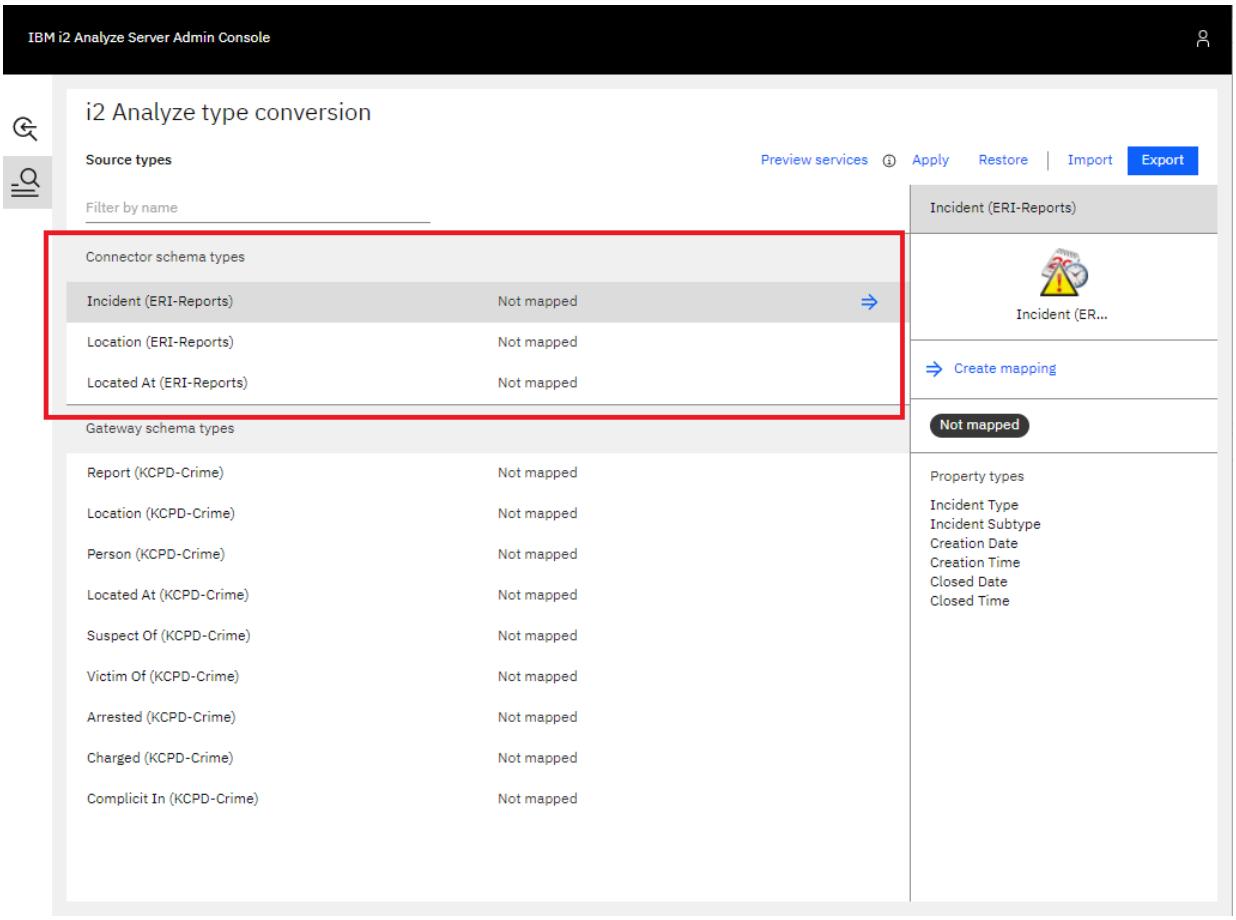
Looking at all the available item types, you can see that both schemas have their own:

- 'Location' entity type
- 'Report'/'Incident' entity type (different names, but modeling the same concept)
- 'Located At' link type

You can use type conversion mappings to resolve this duplication.

Configuring item type mappings from a connector schema to a gateway schema

In a web browser, go to the the **i2 Analyze Type Conversion** app in the i2 Analyze Server Admin Console to see the list of item types that can be mapped. You should see the list of all item types in the ERI-Reports schema, and that none of them has been mapped.

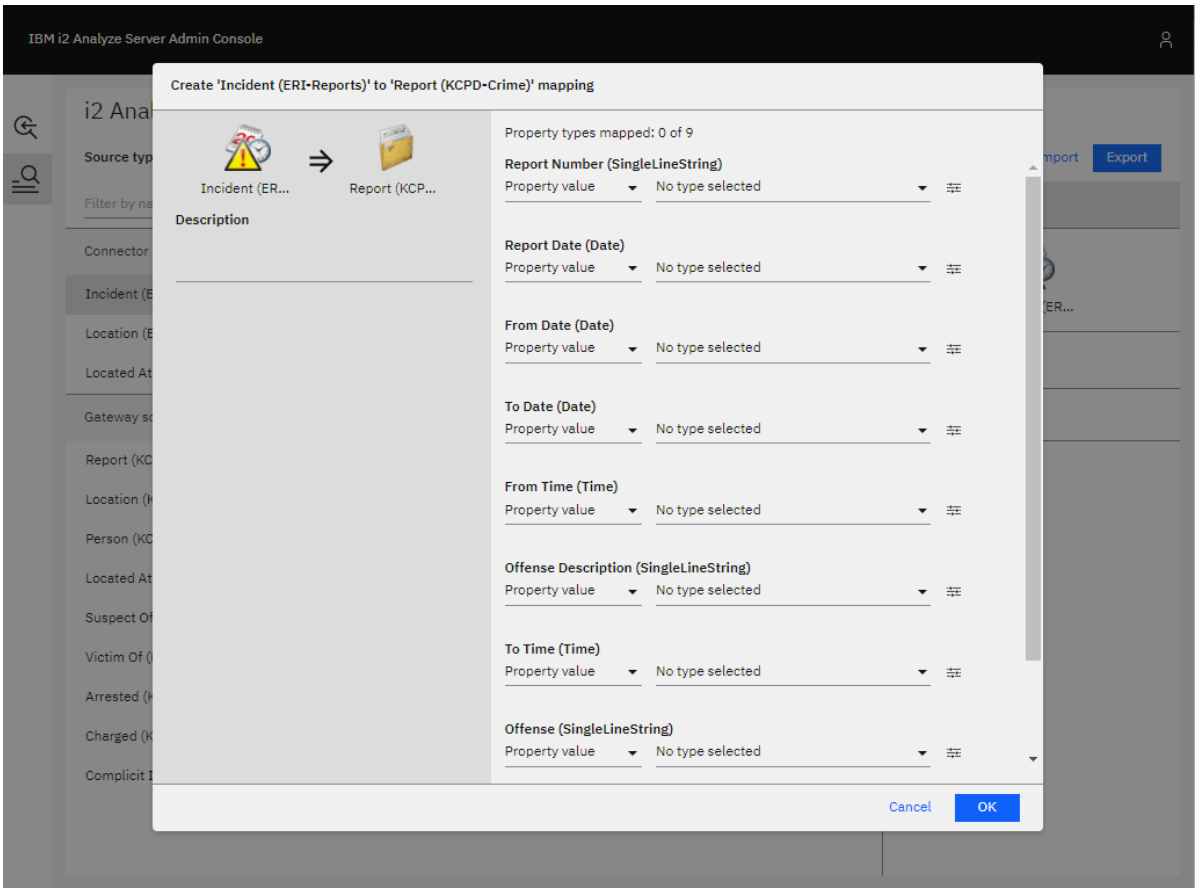


Where appropriate, ERI-Reports item types can be mapped to similar item types in the KCPD-Crime gateway schema.

Incident and Report

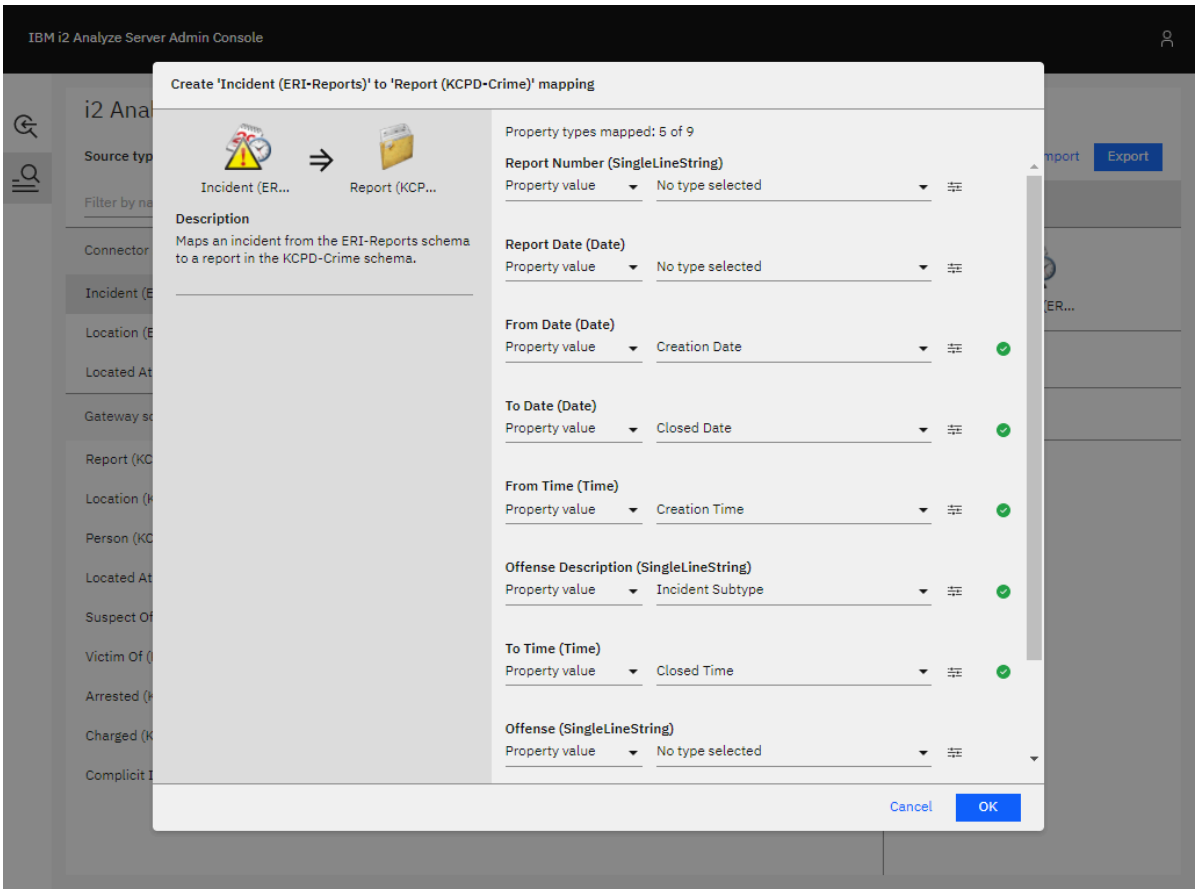
The Incident entity type in the ERI-Reports schema can be mapped to the Report entity type in the KCPD-Crime schema.

1. Select the **Incident (ERI-Reports)** item type in the list, then click **Create mapping** in the right-hand pane. This shows the list of item types to which the Incident item type can be mapped.
2. Select the **Report** item type from the KCPD-Crime schema types, then click **Create mapping**. You will see all the property types of the Report type in the gateway schema, with options to choose how they should be populated when mapping from an ERI-Reports Incident record.



3. In the example below, the property types of the Incident (ERI-Reports) type are mapped to the property types of the Report (KCPD-Crime) according to the following table.

Incident (ERI-Reports)	**Report (KCPD-Crime)**
Creation Date	From Date
Closed Date	To Date
Creation Time	From Time
Closed Time	To Time
Incident Subtype	Offence Description

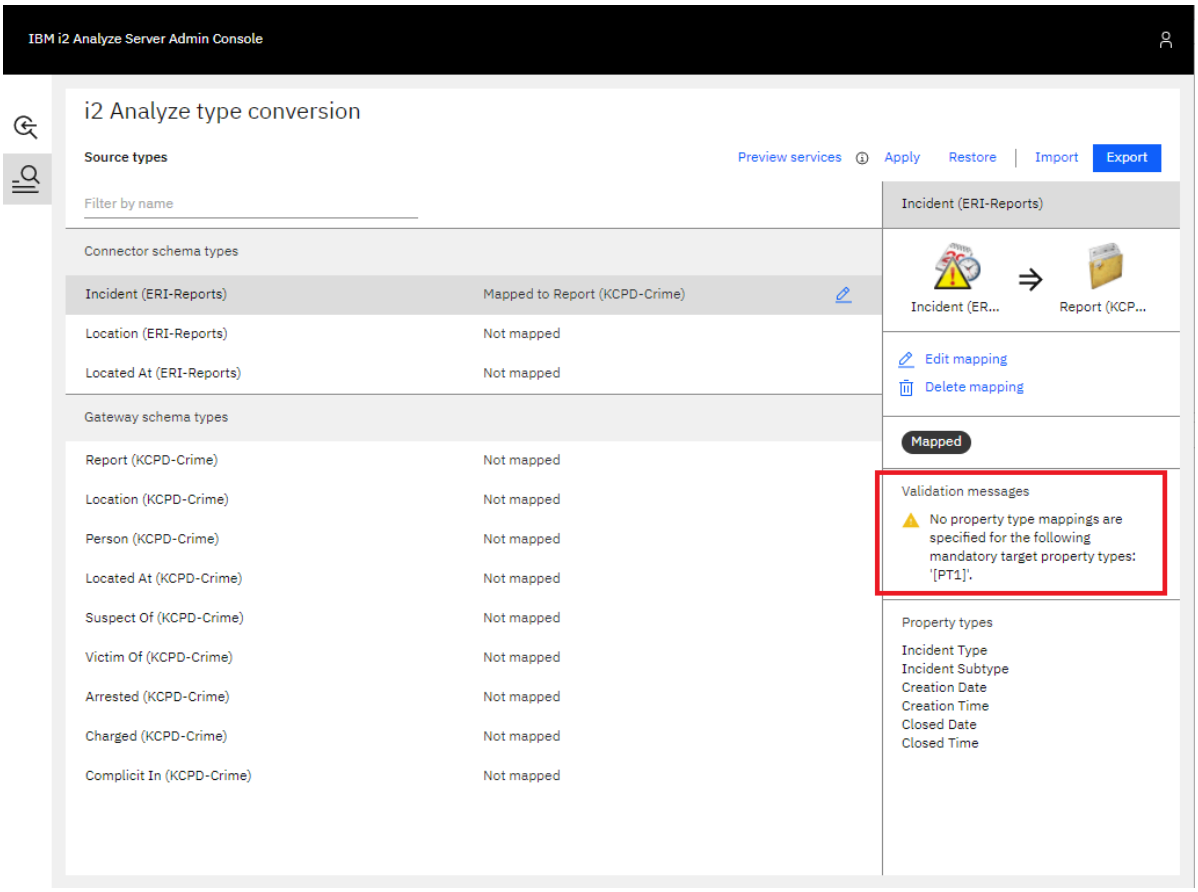


There is no comparable property type in the Incident (ERI-Reports) type for these Report (KCPD-Crime) property types:

- Report Number
- Report Date
- Offense
- Domestic Violence

As a result, these types are left unmapped. They will not be populated in Report records that are converted from Incident records.

4. When you are satisfied with all the property mappings, confirm the mapping by clicking **OK**. The result is a validation warning message that occurs when a mandatory property of the target type is unmapped, which is the case for the mandatory Report Number property type of the Report (KCPD-Crime) item type. Any Report entity created from an Incident entity using this mapping would not have a Report Number, which would make that Report invalid.

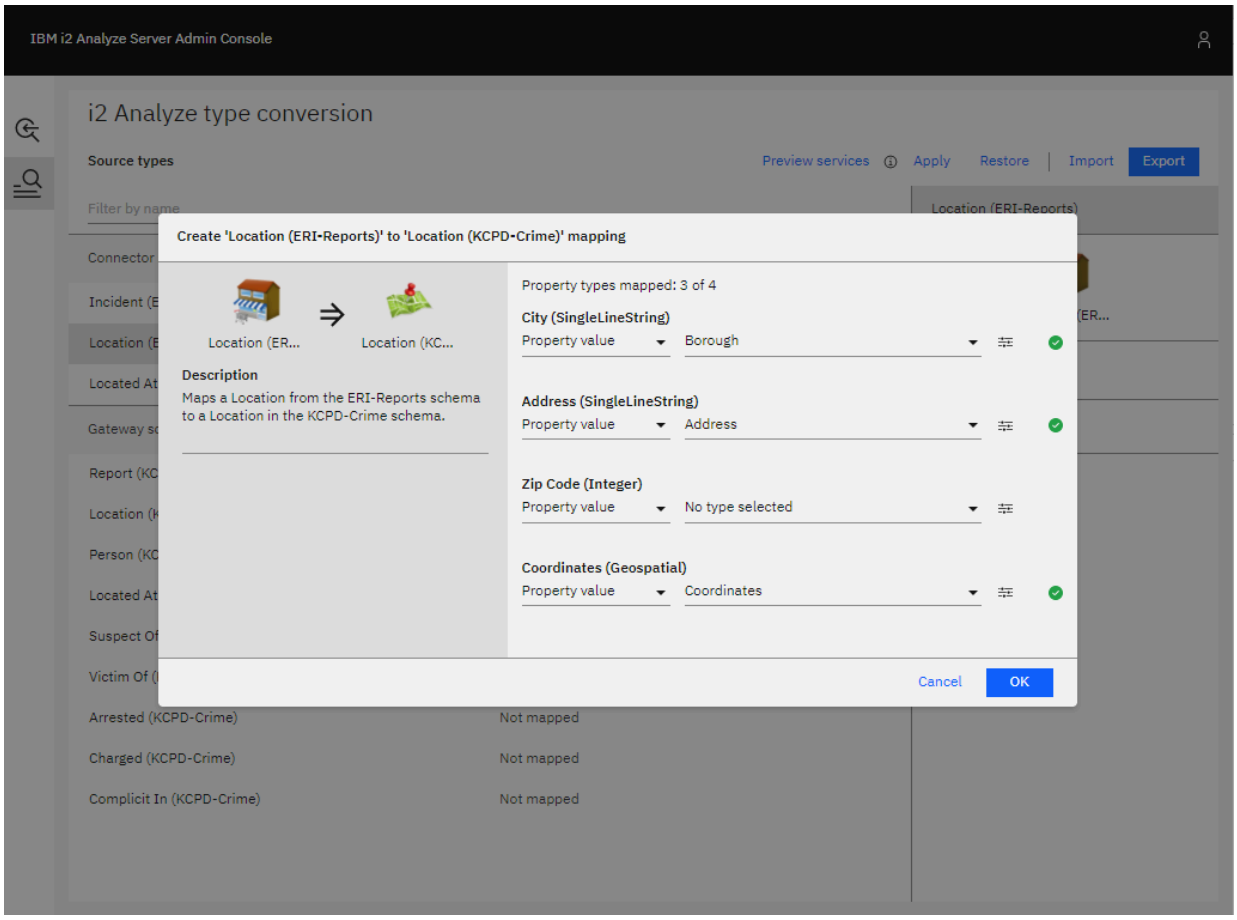


5. To provide a Report Number in all circumstances, select the **Incident (ERI-Reports)** item type in the list, then click **Edit mapping** in the right-hand pane to update the mapping. In the mandatory Report Number (KCPD-Crime) property type, select the dropdown menu and change the type to `Fixed value`. Populate the field with a constant value.
6. After you update the property type mapping, confirm the item type mapping by clicking **OK**.

Location

Following the same process, you can map the Location type from the ERI-Reports schema to the Location type in the KCPD-Crime schema.

The Address and Coordinates property types of Location (ERI-Reports) are automatically mapped to the Address and Coordinates property types of Location (KCPD-Crime). Below is an example configuration you could use.



Again, when you are satisfied with the mapping configuration for Location records, click **OK** to confirm the mapping.

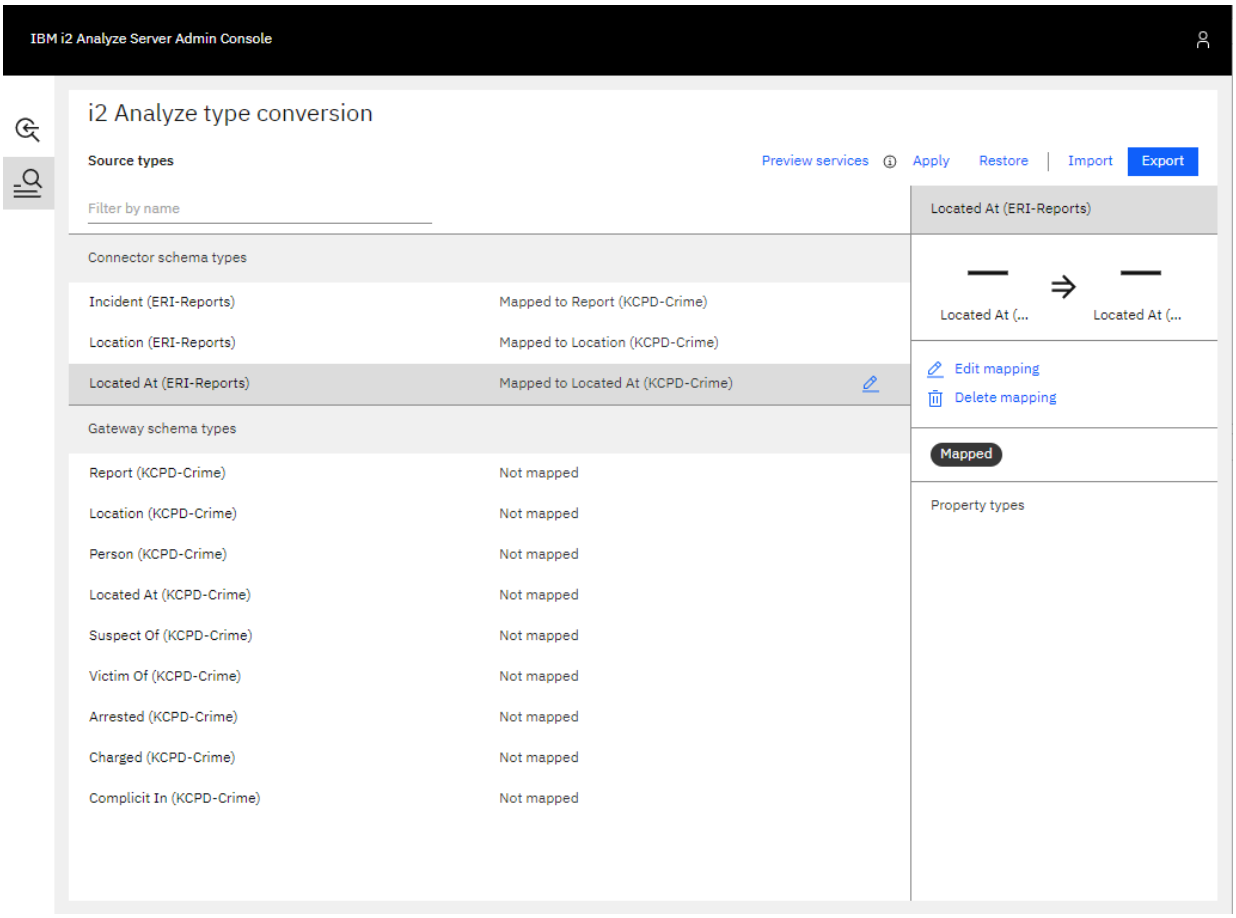
Located At

You can map the Located At (ERI-Reports) link type to the Located At (KCPD-Crime) link type by following the same process:

1. Select **Located At (ERI-Reports)** in the list of types, then click **Create mapping** in the right-hand pane.
2. Select **Located At** from the KCPD-Crime schema types, then click **Create mapping**.
3. There are no property types to map, so just add a description.
4. Confirm the mapping by clicking **OK**.

Testing the item type mappings

When you have defined the item type mappings, you should see an updated list of types like the example shown below.



To test the item type mappings by previewing the external searches provided by the KCPD and ERI connectors:

1. Click **Apply** in the top-right. This applies the mappings to the test environment that is available only through the Admin Console. It does not apply the mappings to the live server.
2. Click **Preview services** to open a preview of how the services would behave with the mappings you have configured. Notice how the ERI Connector's "All data" service now returns item types from the KCPD-Crime schema.
3. Go back and make any changes to the mappings, repeating steps 1 and 2 until you are satisfied with the configuration.

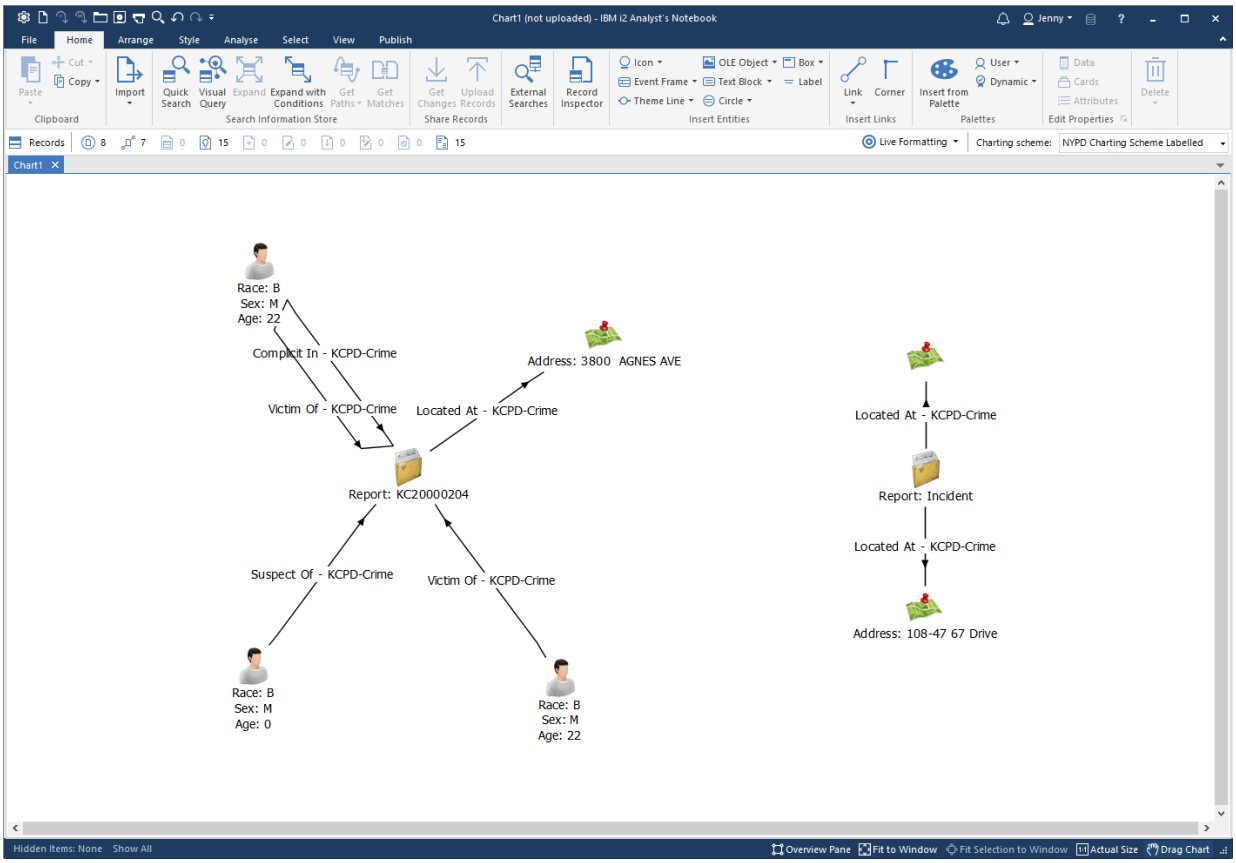
Applying the item type mappings to the i2 Analyze server

To apply the mapping configuration you have created on the i2 Analyze server for all users, see Applying the mapping configuration to the i2 Analyze server.

The result

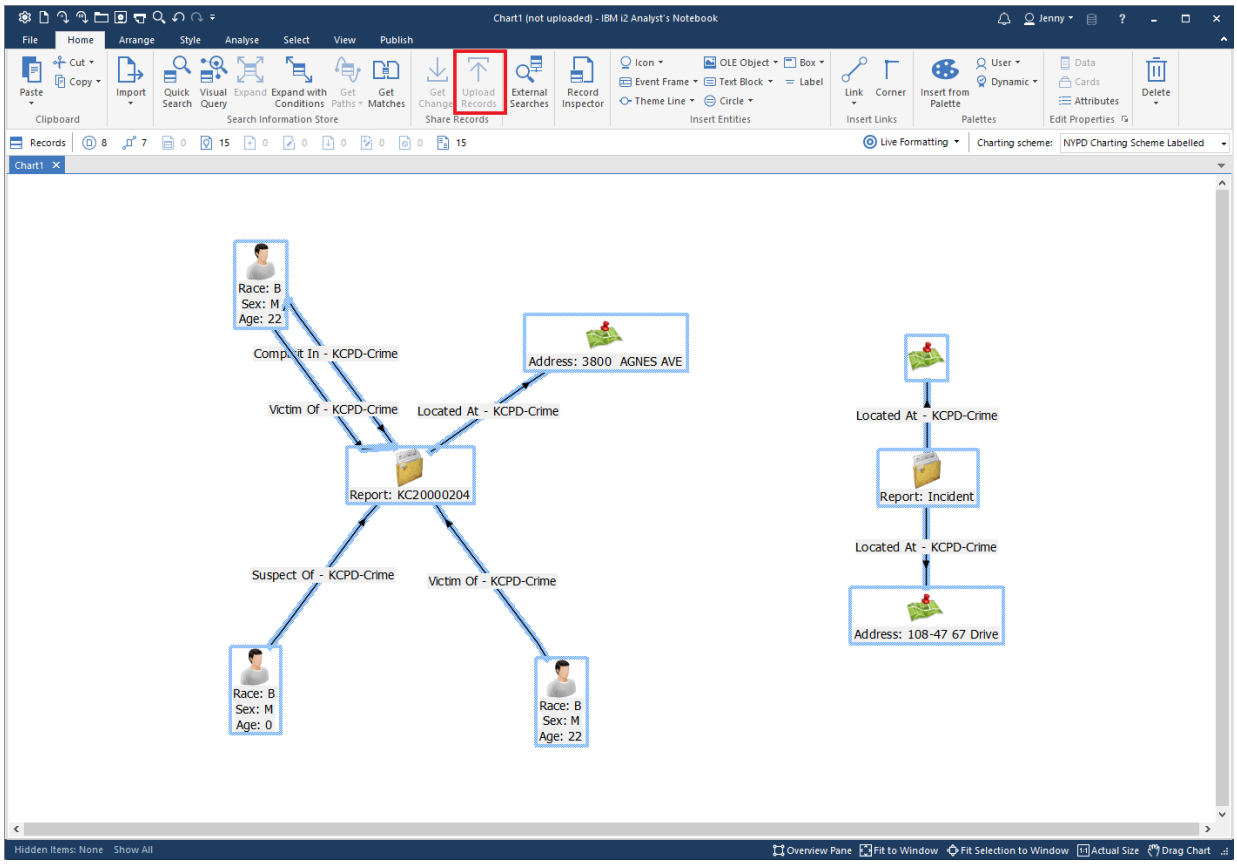
In Analyst's Notebook, use some of the services provided by the connectors and copy some results from each connector to your chart. Notice how there are now no duplicate item types.

For example, compare the chart below to the one at the very beginning of this walkthrough. You can see that all of the records have types from the KCPD-Crime schema, including the records from the ERI connector. This is because of the mapping configuration that maps the item types in the ERI-Reports schema to the item types in the KCPD-Crime schema.



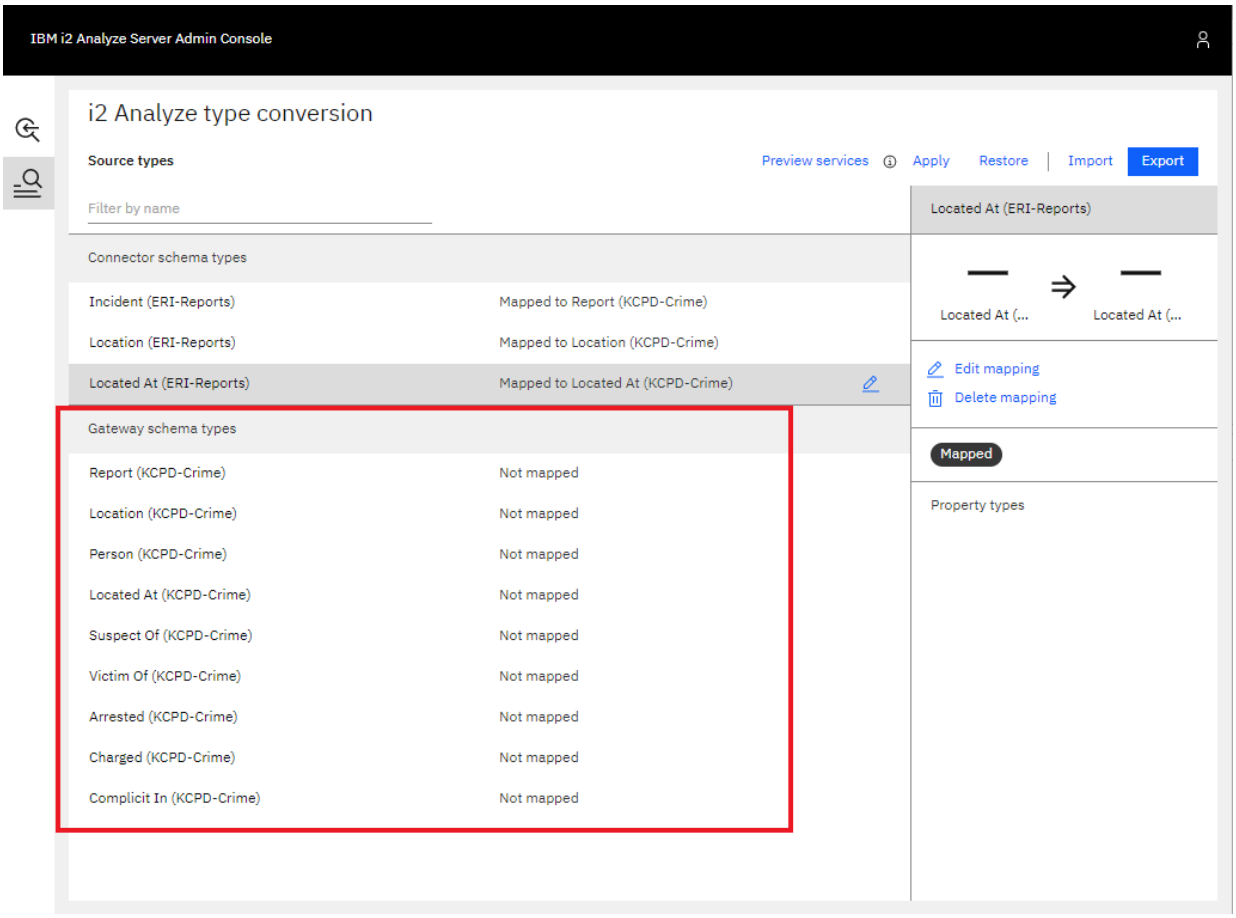
Uploading converted records to the Information Store

Mapping connector types to gateway types does not result in records that can be uploaded to the Information Store. The item types of the records are defined in the KCPD-Crime gateway schema, not the Information Store schema. Next, you can define mappings from the gateway schema to the Information Store schema, and see that the records returned from both connectors can then be uploaded to the Information Store.



Configuring item type mappings from a gateway schema to an Information Store schema

In a web browser, go to the **i2 Analyze Type Conversion** app in the i2 Analyze Server Admin Console to see the list of item types that can be mapped. You should see the list of all item types in the KCPD-Crime schema, and that none of them has been mapped.



Where appropriate, KCPD-Crime item types can be mapped to similar item types in the NYPD-Complaints Information Store schema.

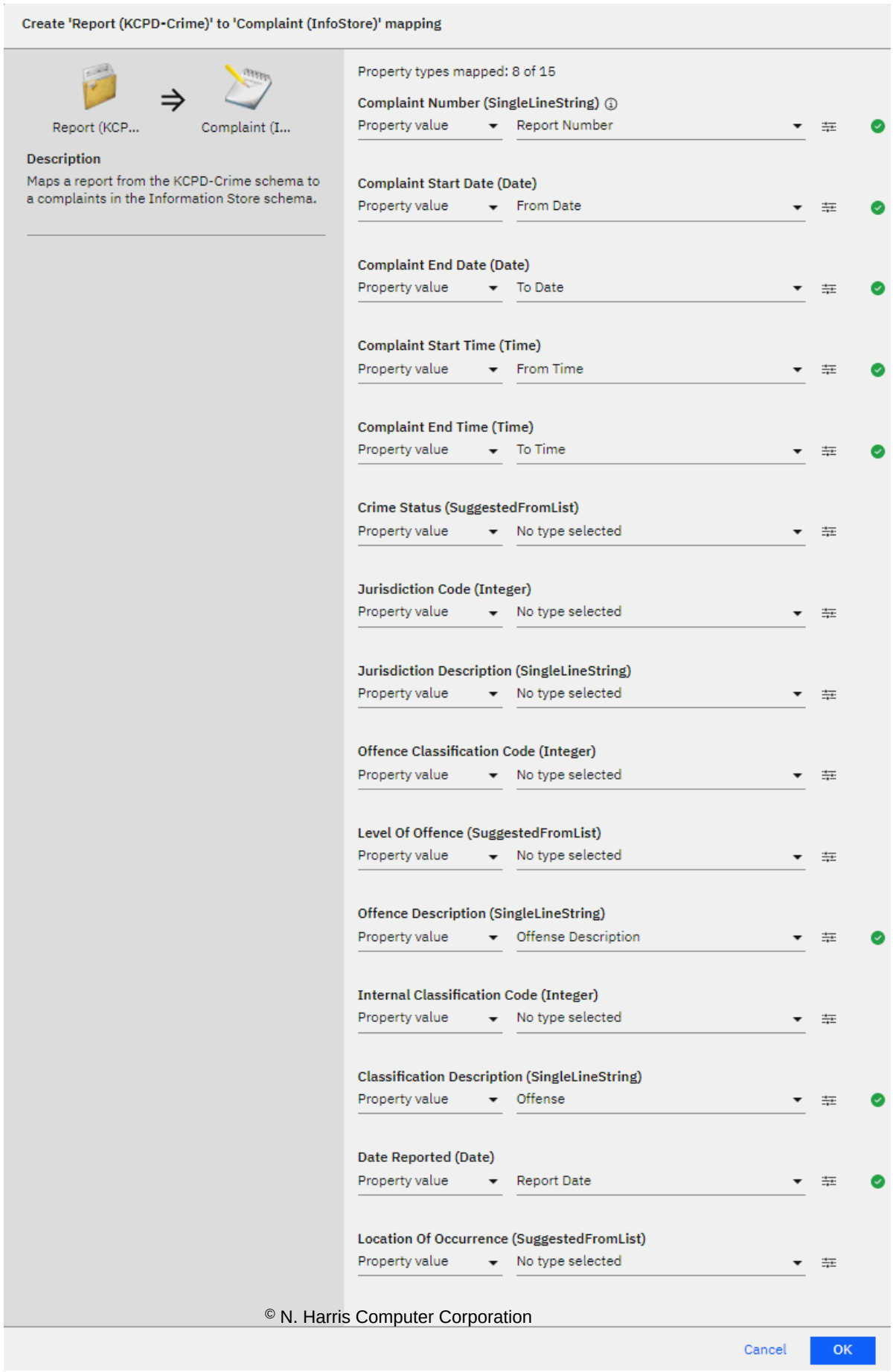
Report and Complaint

The Report entity type in the KCPD-Crime schema can be mapped to the Complaint entity type in the Information Store schema.

1. Select the **Report (KCPD-Crime)** item type in the list, then click **Create mapping** in the right-hand pane. This shows the list of item types to which the Report item type can be mapped.
2. Select the **Complaint** item type from the Information Store schema, then click **Create mapping**. You will see all the property types of the Complaint type in the Information Store schema, with options available to choose how they should be populated when mapping from a KCPD-Crime Report record.
3. In the example below, the property types of the Report (KCPD-Crime) type are mapped to the property types of the Complaint (InfoStore) according to the following table.

Report (KCPD-Crime)	**Complaint (InfoStore)**
Report Number	Complaint Number

Report (KCPD-Crime)	**Complaint (InfoStore)**
From Date	Complaint Start Date
To Date	Complaint End Date
From Time	Complaint Start Time
To Time	Complaint End Time
Offense Description	Offence Description
Offense	Classification Description
Report Date	Date Reported



There is no comparable property type in the Report (KCPD-Crime) type for the Complaint (InfoStore) property types:

- Crime Status
- Jurisdiction Code
- Jurisdiction Description
- Offence Classification Code
- Level Of Offence
- Internal Classification Code
- Location Of Occurrence

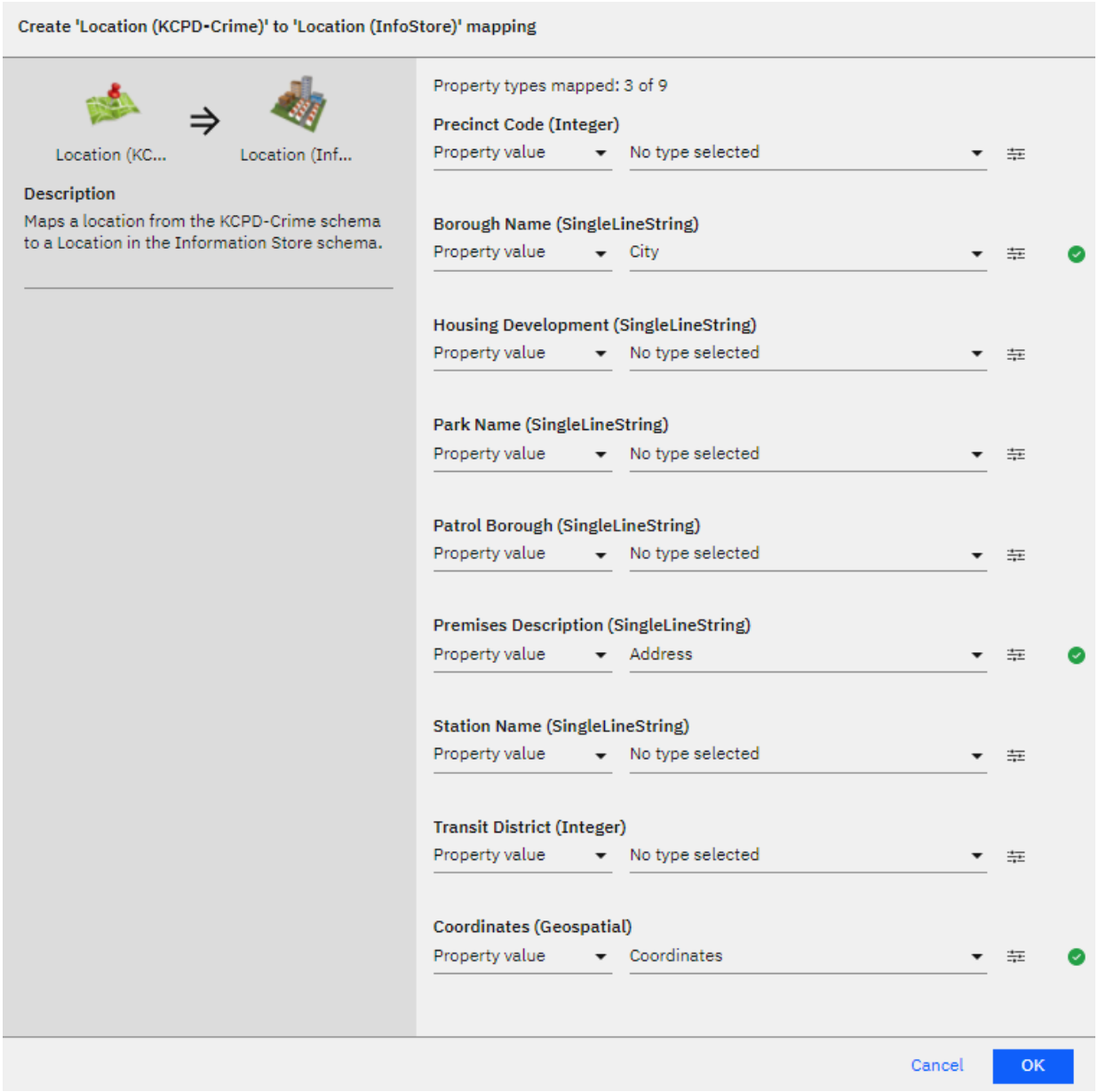
As a result, these types are left unmapped. The properties of Complaints records that are converted from Report records will not be populated.

4. When you are satisfied with all the property type mappings, confirm the mapping by clicking **OK**.

Location

Following the same process, the Location type from the KCPD-Crime schema can be mapped to the Location type in the Information Store schema.

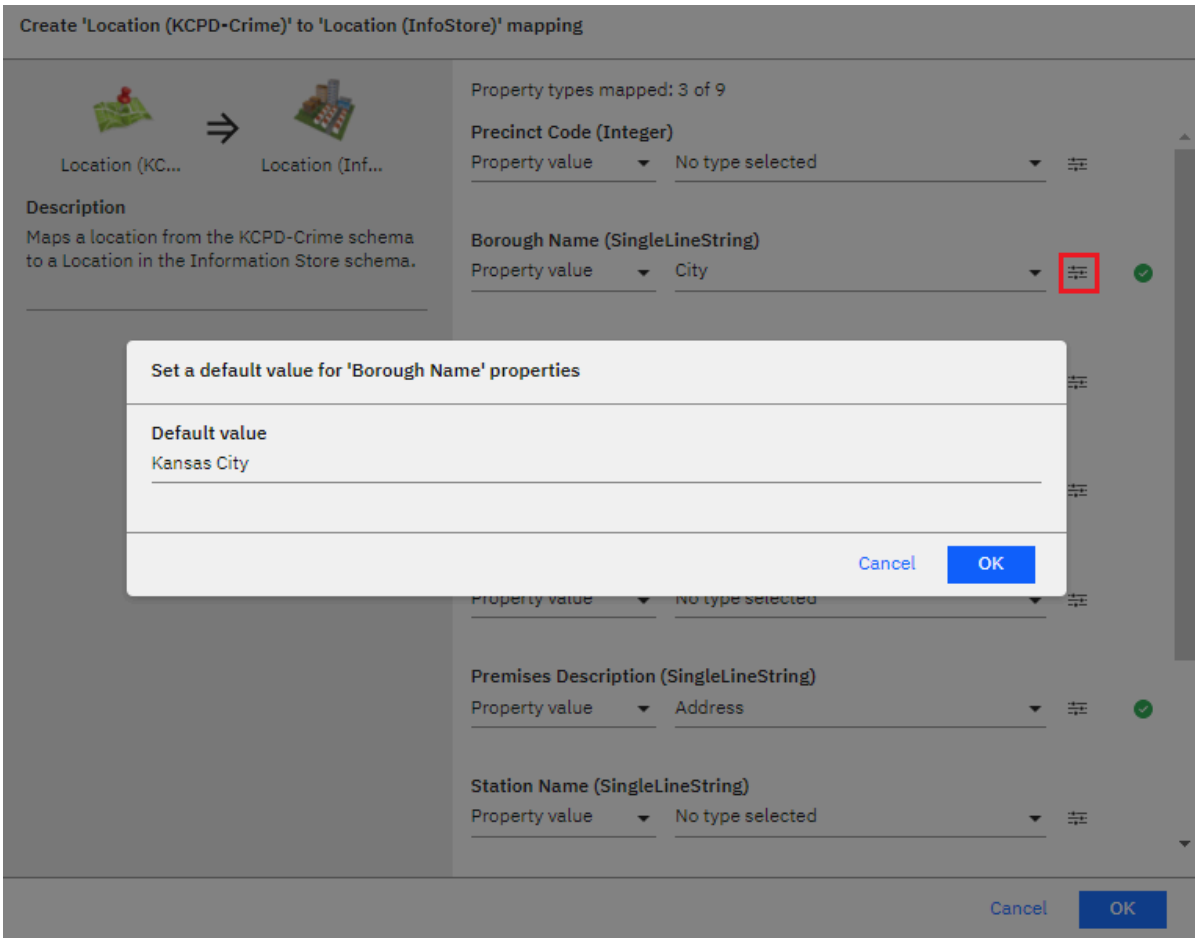
The Coordinates property type of Location (KCPD-Crime) is automatically mapped to the Coordinates property type of Location (InfoStore). The image shows an example configuration that you could use.



You can see that the Borough Name properties of Information Store Locations will be populated from the value of the City properties of KCPD-Crime Location records. These property types are not an exact match for one another, but this mapping at least makes it clear when Locations are in Kansas City rather than New York City, in case Coordinates are not provided.

You can go further and ensure the Borough Name property is always populated in converted Locations by setting a default value that will be used if the source KCPD-Crime Location record does not contain a value for the City property. To do this:

1. Click the button highlighted to the right of the Borough Name configuration shown below.



2. Enter a default value to use for the Borough Name property if a source KCPD-Crime Location record has no City value. For example, "Kansas City".
3. Click **OK**.

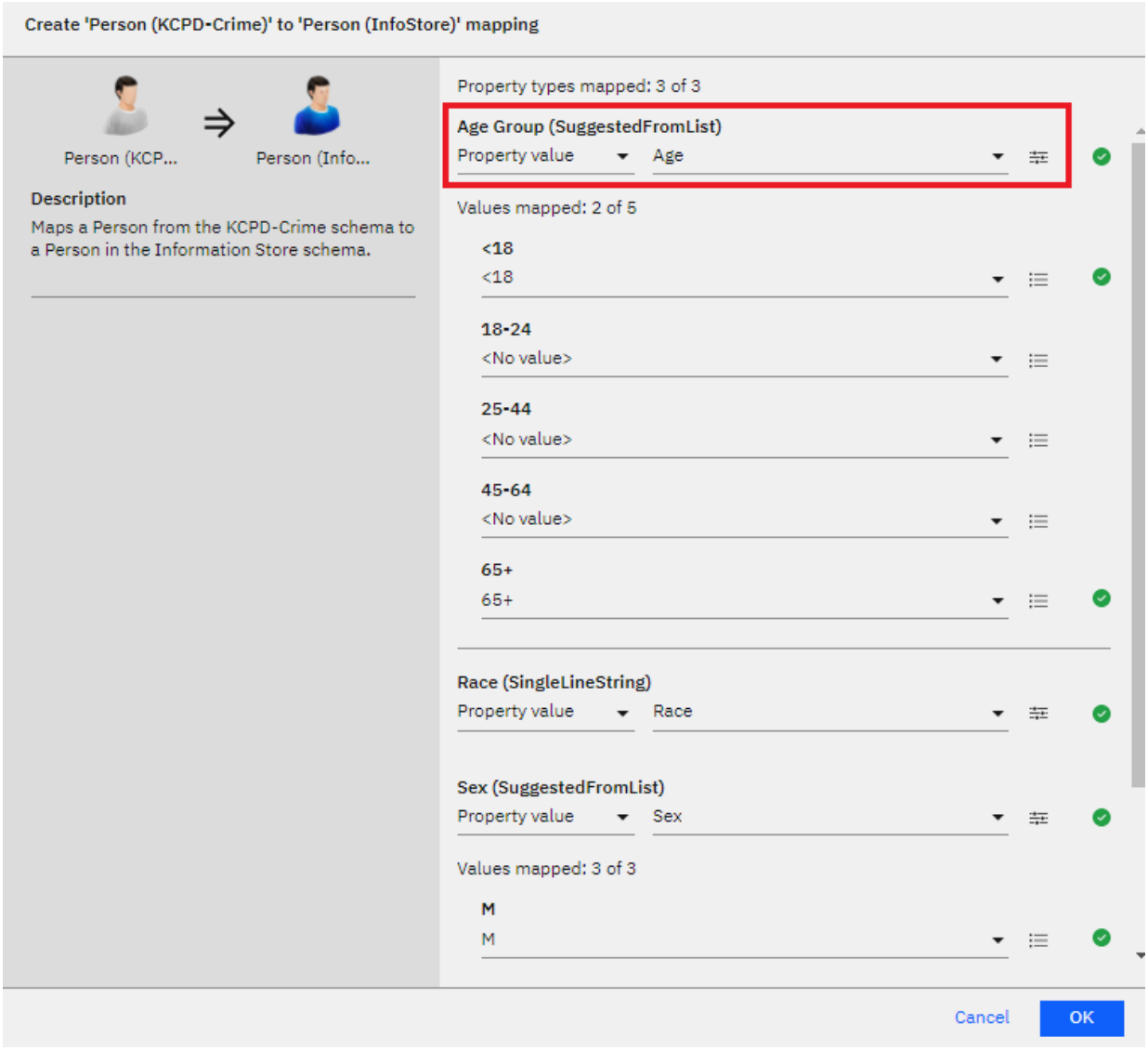
The mapping of Address to Premises Description does not seem like a perfect match either. But the Address of a Location might be too important not to have, and the Premises Description is a suitable target property in which to store it.

Again, when you are satisfied with the mapping configuration for Location records, click **OK** to confirm the mapping.

Person

You can follow the same process to create a mapping from the Person (KCPD-Crime) type to the Person (InfoStore) type.

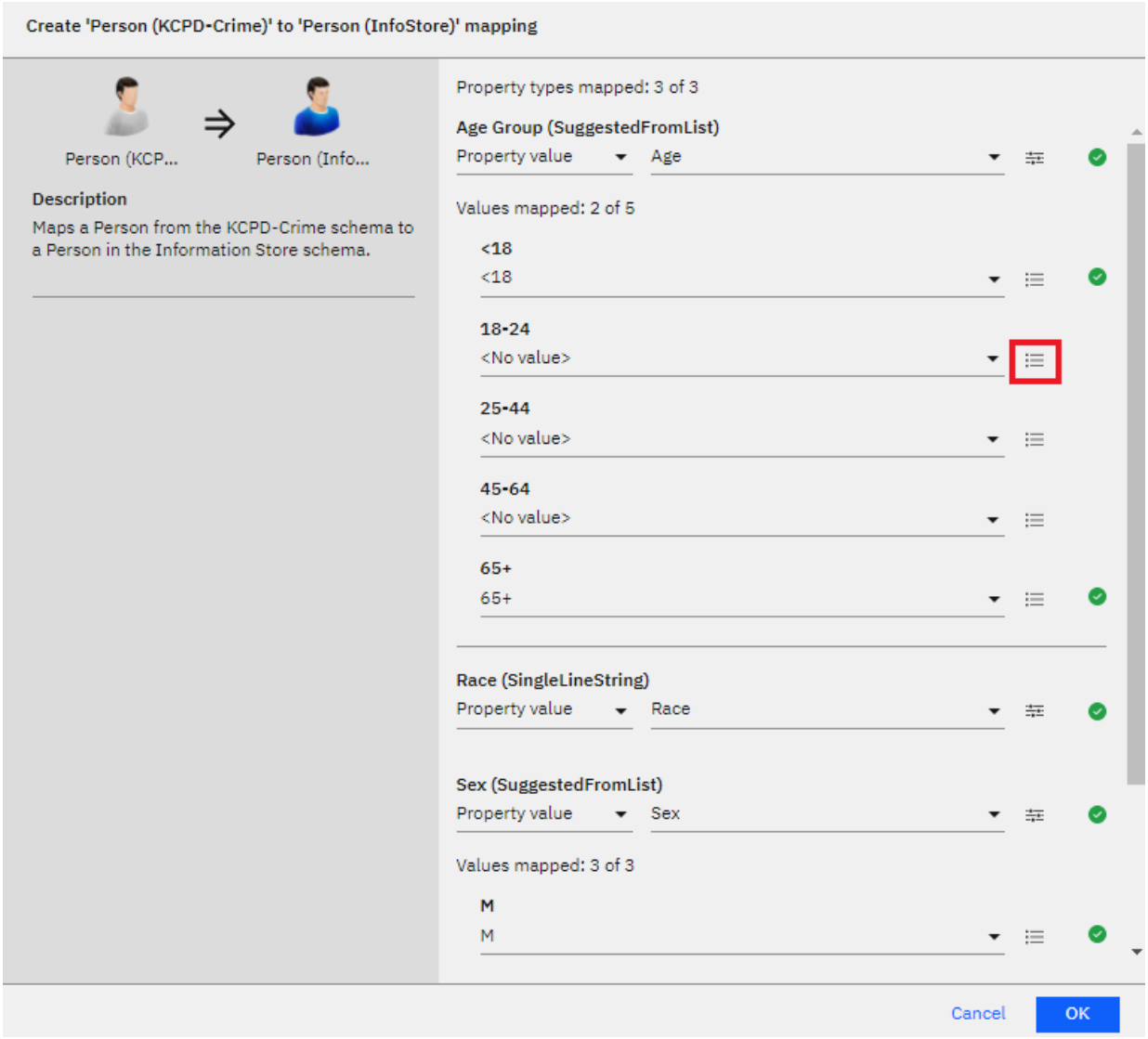
The Race and Sex property type mappings are generated automatically, so the only property type left is Age Group. You can map the values of the Age property type from the Person (KCPD-Crime) type, as shown below.



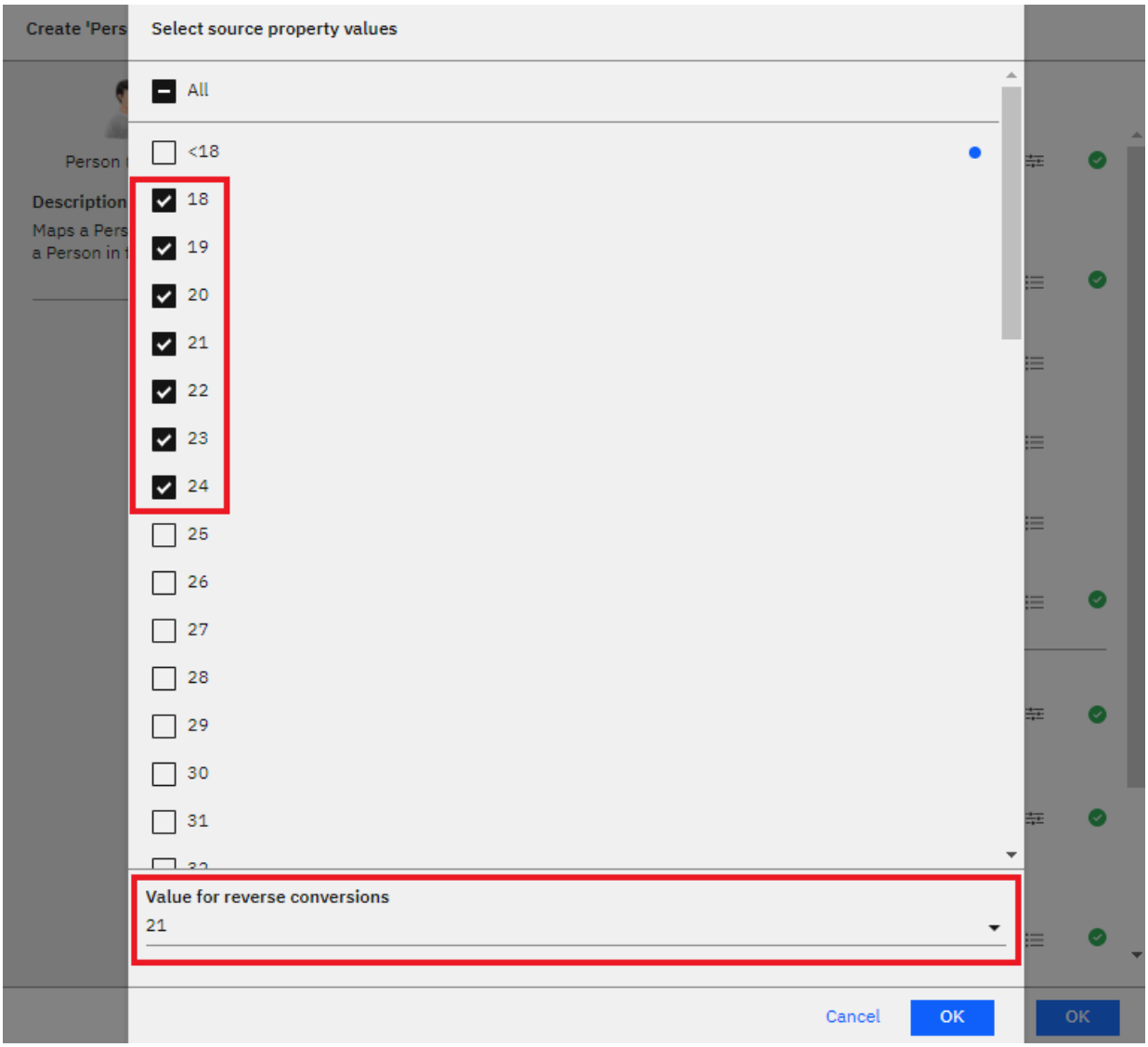
The Age property types of the source type and the target type are both `SUGGESTED_FROM` types, but they have different suggested values. The source property type, from the KCPD-Crime schema, has suggested values: `<18`, `19`, `20`, `21`, ..., `64`, and `65+`; whereas the target property type, from the Information Store schema, has suggested values: `<18`, `18-24`, `25-44`, `45-64`, and `65+`.

The `<18` and `65+` values are automatically mapped to one another. For the remaining target values (`18-24`, `25-44`, and `45-64`), you must choose which source values should map to them.

You have the option to select a single value of the source property type from the dropdown, but that wouldn't make much sense. The source property values `18`, `19`, `20`, `21`, `22`, `23`, and `24` should *all* map to the `18-24` target value. To do this, click the button to the right of the dropdown menu beneath *18-24*, as highlighted below.



Then, select all the appropriate values as shown below, and select a source value to use when the conversion is applied in reverse during any seeded searches.



Located At

The Located At (KCPD-Crime) link type can be mapped to the Located At (InfoStore) link type by following the same process:

1. Select **Located At (KCPD-Crime)** in the list of types, then click **Create mapping** in the right-hand pane.
2. Select **Located At** from the Information Store link types, then click **Create mapping**.
3. There are no properties to map, so just add a description if you wish.
4. Confirm the mapping by clicking **OK**.

Suspect Of

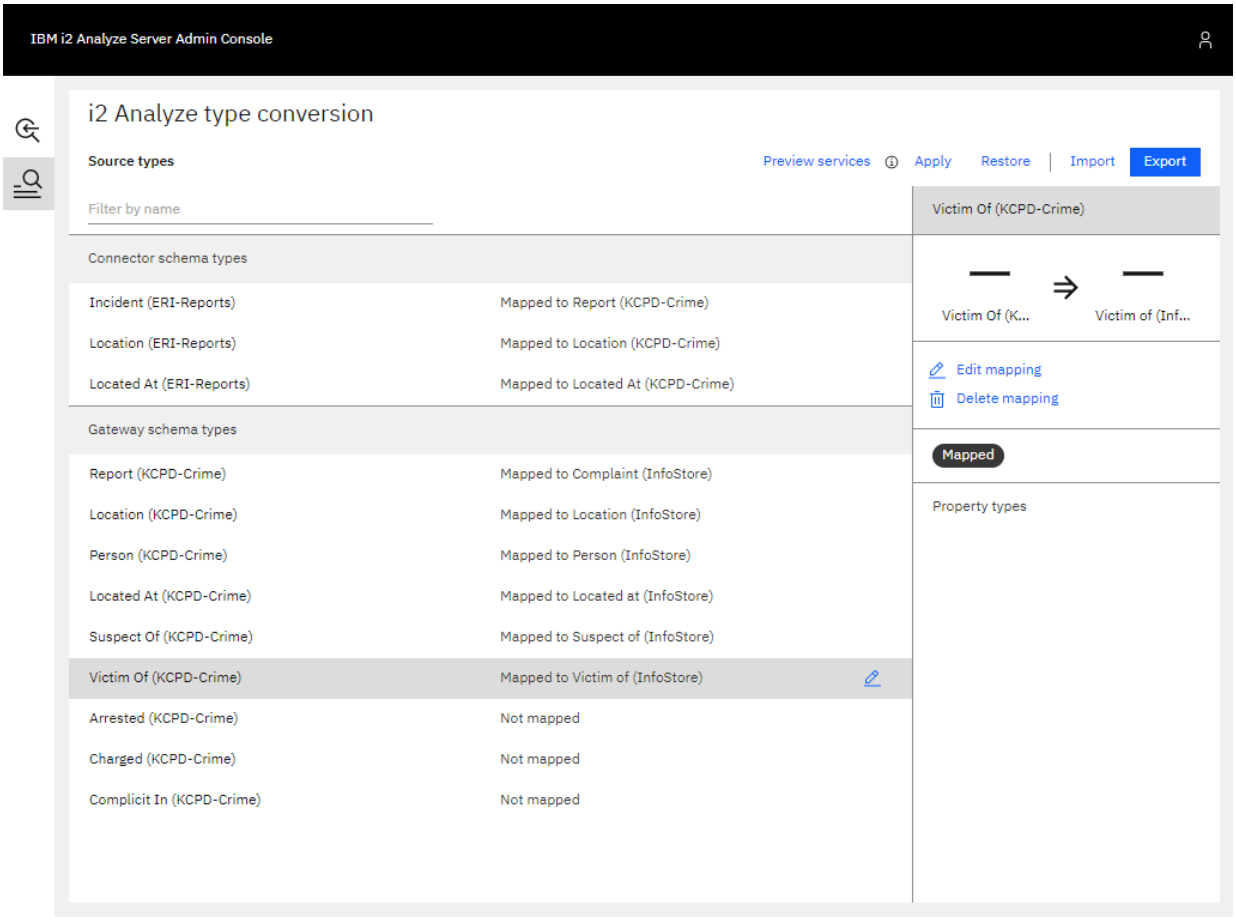
Map Suspect Of (KCPD-Crime) to Suspect Of (InfoStore) in the same way as Located At.

Victim Of

Map Victim Of (KCPD-Crime) to Victim Of (InfoStore) in the same way as Located At.

Testing the item type mappings

When you have defined the item type mappings, you should see an updated list of types like the example shown below.



The Arrested, Charged, and Complicit In link types are left unmapped, because there are no suitable target link types in the existing Information Store schema. i2 Analyze will recognize that their end types have been mapped, and allow the Information Store schema's Person and Complaint types to be connected by these links.

To test the item type mappings by previewing the external searches provided by the KCPD and ERI connectors:

1. Click **Apply** in the top-right. This applies the mappings to the test environment that is available only through the Admin Console. (It does not apply the mappings to the live server.)
2. Click **Preview services** to open a preview of how the services would behave with the mappings you have configured. Notice how the KCPD Connector's Get All service and ERI Connector's All data service now return item types from the Information Store schema.

3. Go back and make any changes to the mappings, repeating steps 1 and 2 until you are satisfied with the configuration.

Applying the item type mappings to the i2 Analyze server

To apply the mapping configuration you have created on the i2 Analyze server for all users, see Applying the mapping configuration to the i2 Analyze server.

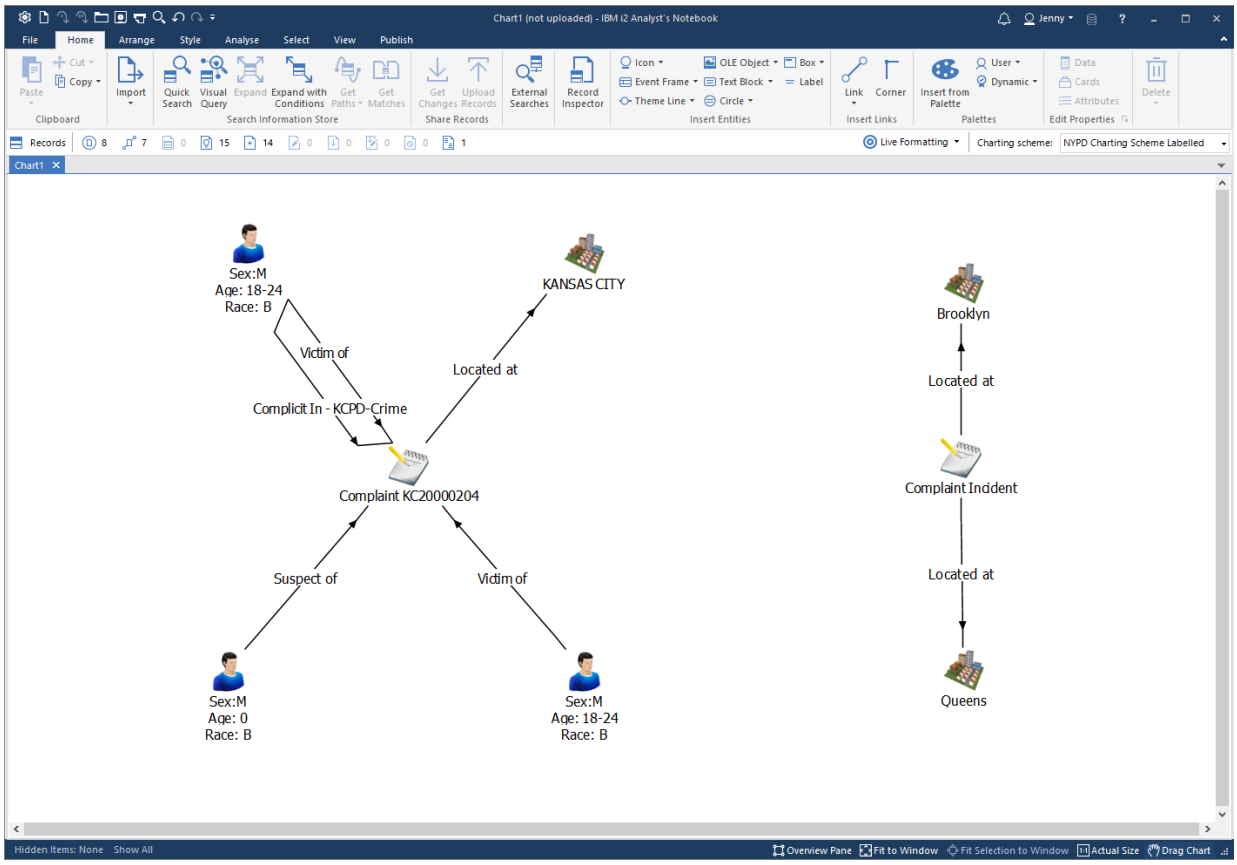
The result

In Analyst's Notebook, use some of the services provided by the connectors and copy some results from each connector to your chart. Notice how the KCPD and ERI connectors now return records whose item types are defined in the Information Store schema.

The KCPD connector returns records whose item types are defined in the KCPD-Crime schema, which are converted directly to records whose item types are defined in the Information Store schema.

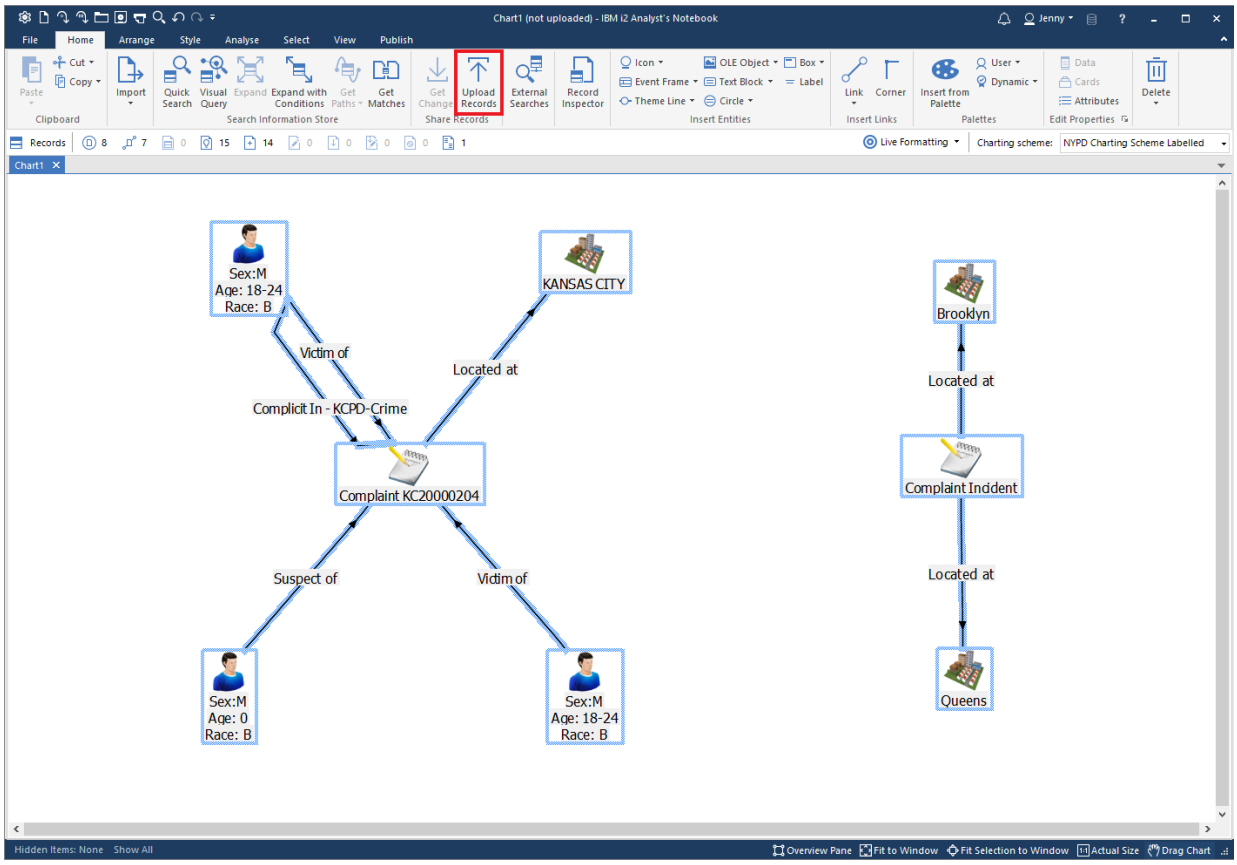
The ERI connector returns records whose item types are defined in the ERI-Reports schema. These types are mapped to item types in the KCPD-Crime schema, which are themselves mapped to item types defined in the Information Store schema. What happens as a result is that the records returned by the ERI connector are converted to records that align with the Information Store schema.

For example, compare the chart below to the one at the beginning of this walkthrough. You can see that all of the records have types from the Information Store schema, except for the unmapped Complicit In link in the top-left corner.

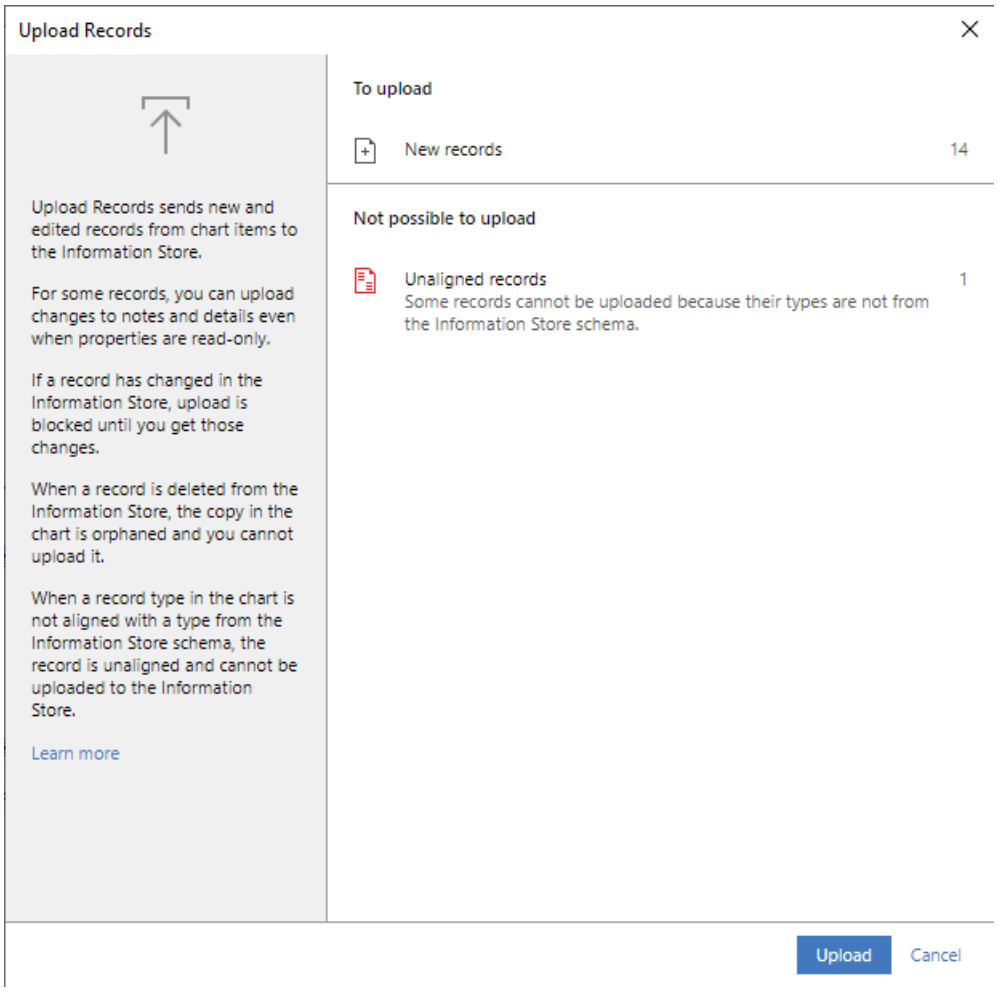


Uploading converted records to the Information Store

If you want to upload the selected records to the Information Store, you will notice that the button is now enabled.



Click **Upload Records**. In the open window, you will see that one of the records - the unmapped Complicit In link in the top-left corner - cannot be uploaded because its item type is not from the Information Store schema.



If you want to upload this record, you must configure a mapping for its item type. Otherwise, you can click **Upload** to upload the aligned records to the Information Store.

Merging two schemas

In this example of how to create type conversion mappings, two connectors - each with their own gateway schema - are being used in an i2 Analyze deployment. The two schemas contain similar item types, so item type mappings will be used to remove duplicates.

It is not possible to define type conversion mappings in both directions between two schemas. That is, you cannot map item types from schema A to schema B, and also map item types from schema B to schema A. In a situation like this, you can create a third gateway schema that contains all the types you want from the two original schemas, and then define mappings from the originals to the new schema.

Setting up the scenario

To follow this scenario, deploy i2 Analyze with the example NYPD connector and KCPD connector, both configured to use gateway schemas.

Reviewing the item types

Open the `schema/nypd-complaint-data-schema.xml` and `schema/kcpd-crime-data-schema.xml` schemas in Schema Designer to see the item types they define. Both schemas contain entity types that represent:

- people
- complaints/reports
- locations

Both schemas also contain link types that represent:

- a complaint having occurred at a location
- a person being the victim of a complaint
- a person being the suspect of a complaint

The KCPD-Crime schema contains additional link types that represent:

- a person being complicit in a complaint
- a person having been arrested for a complaint
- a person having been charged for a complaint

There are obviously duplicate item types that can be resolved by defining item type mappings. Review the item types and decide which of the duplicate types you prefer. For the purposes of this example, the preferred item types are:

- Person (NYPD-Complaints)
- Report (KCPD-Crime)
- Location (KCPD-Crime)
- Located At (NYPD-Complaints)
- Suspect Of (NYPD-Complaints)
- Victim Of (NYPD-Complaints)

However, it is not possible to map the Person type in the KCPD-Crime schema to the Person type in the NYPD-Complaints schema, *and* map the Complaint type in the NYPD-Complaints schema to the Report type in the KCPD-Crime schema. To resolve this, you can create a schema that contains the preferred types from both schemas, configure it as a gateway schema, and then map types from the NYPD-Complaints schema and the KCPD-Crime schema to it.

Creating a schema

Use Schema Designer to create a schema that contains types identical to the preferred item types you chose from the NYPD-Complaints and KCPD-Crime schemas. An example containing the types listed above is provided as `schema/nypd-kcpd-merged-schema.xml` in the analyze-connect repository.

Configure the new schema as a gateway schema in your deployment. Choose an appropriate short name for this schema, such as "NYPD-KCPD-Merged".

Configuring item type mappings

Follow the process outlined in Configuring item type mappings to define mappings of types from the NYPD-Complaints and KCPD-Crime schemas to types in the NYPD-KCPD-Merged schema. Examples of specific mappings you might define are outlined below, but the general idea is to map pairs of

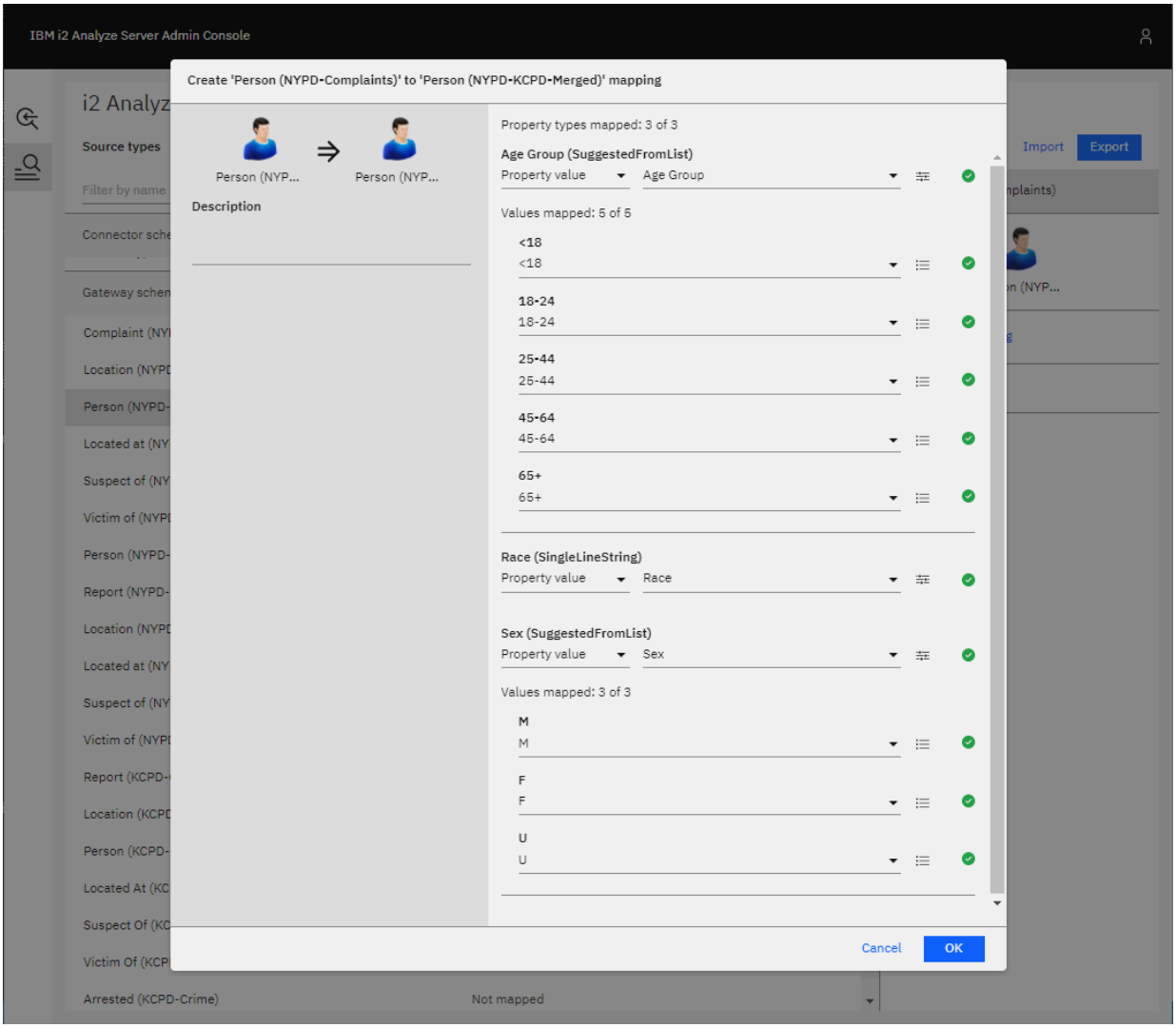
duplicate types in NYPD-Complaints and KCPD-Schema to their corresponding type in the new merged schema.

Start by opening the i2 Analyze Server Admin Console.

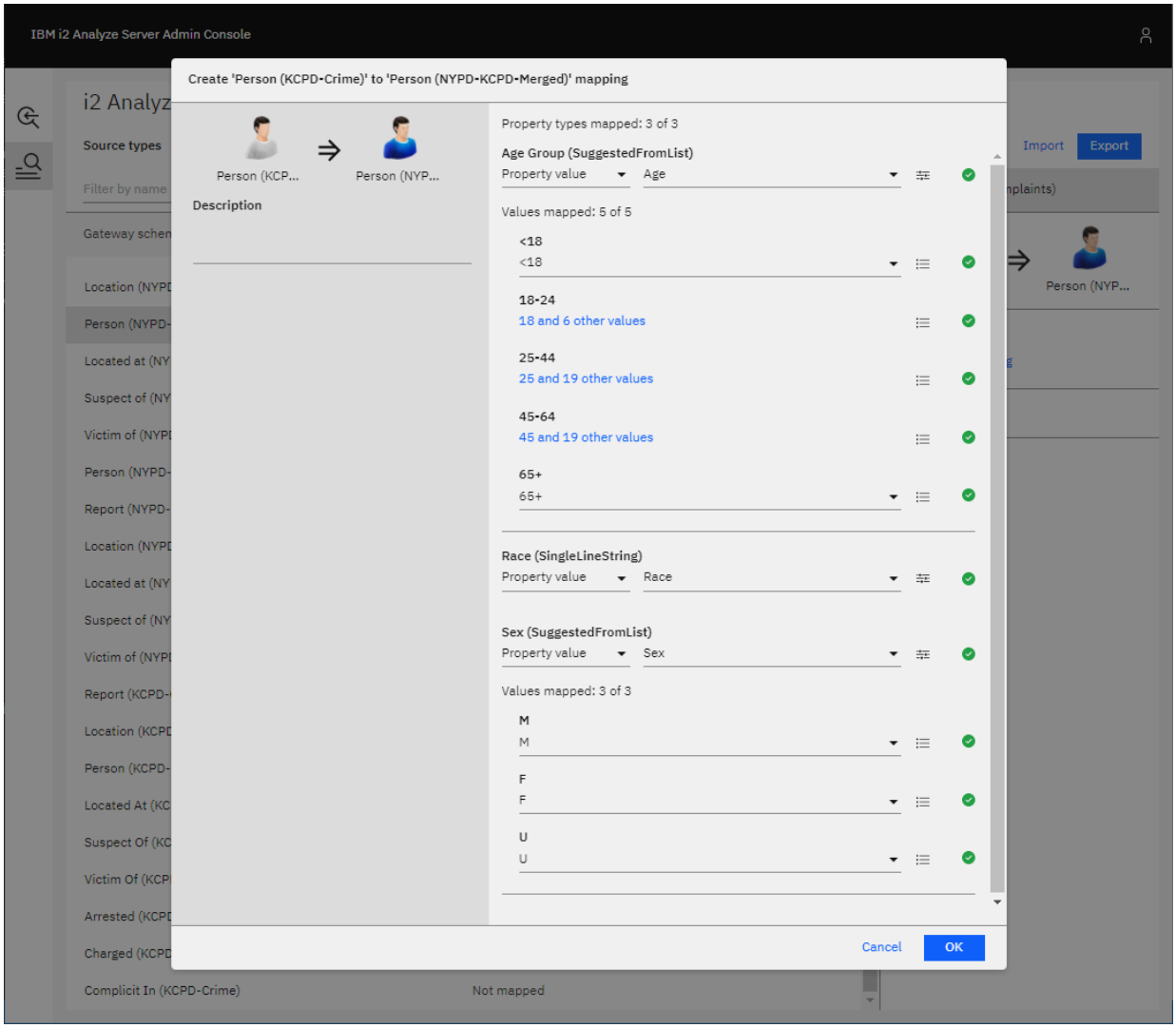
Person

The preferred Person type listed above is the one defined in the NYPD-Complaints schema. This is duplicated as the Person type in the new NYPD-KCPD-Merged schema, so you can define mappings from the Person types in the NYPD-Complaints and KCPD-Crime schemas to the Person type in the NYPD-KCPD-Merged schema.

First, create a mapping from Person (NYPD-Complaints) to Person (NYPD-KCPD-Merged). Since these two types are identical, all the properties will have mappings generated automatically, as shown below. You can just click **OK** to confirm the mapping.



Second, create a mapping from Person (KCPD-Crime) to Person (NYPD-KCPD-Merged) and map the properties as if you were mapping to the NYPD-Complaints Person type. Example property mappings you might define are shown below.

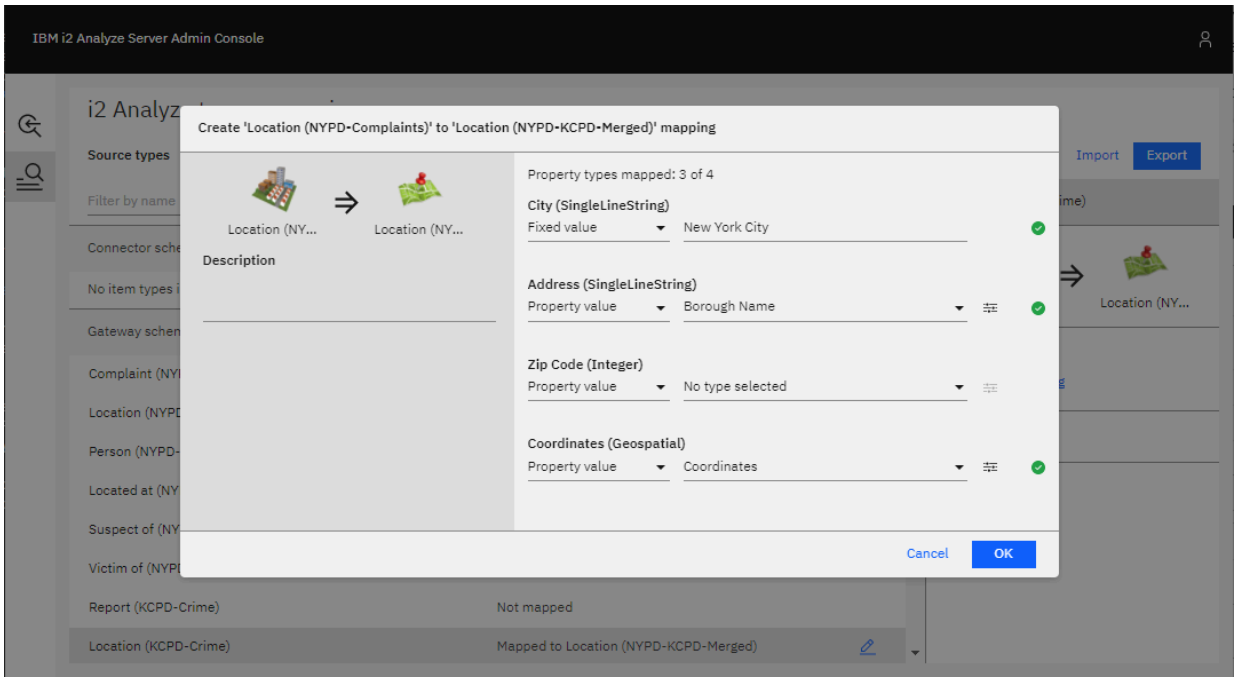


Location

Similarly, the Location (NYPD-Complaints) and Location (KCPD-Crime) types can both be mapped to Location (NYPD-KCPD-Merged).

The Location (NYPD-KCPD-Merged) type is identical to Location (KCPD-Crime), since that is the preferred Location type. Start by mapping Location (KCPD-Crime) to Location (NYPD-KCPD-Merged), making use of the automatically-generated property mappings.

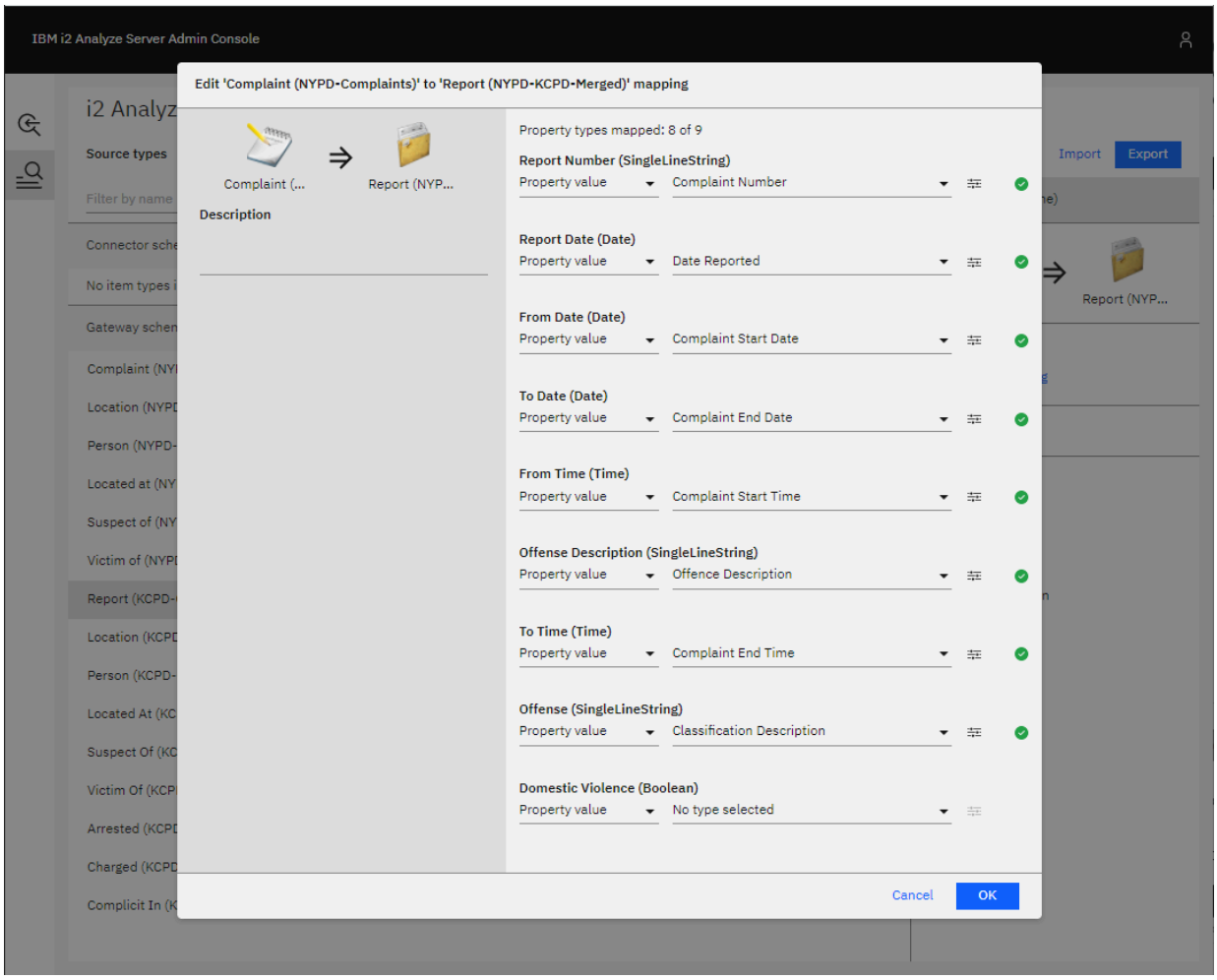
Then, map Location (NYPD-Complaints) to Location (NYPD-KCPD-Merged). Examples of property mappings you might define are shown below.



Report

Following the same process, map the Complaint (NYPD-Complaints) and Report (KCPD-Crime) types to the Report (NYPD-KCPD-Merged) type. The Report type in the merged schema is identical to the Report type in the KCPD-Crime schema, so once again all the property mappings will be populated for you in that case.

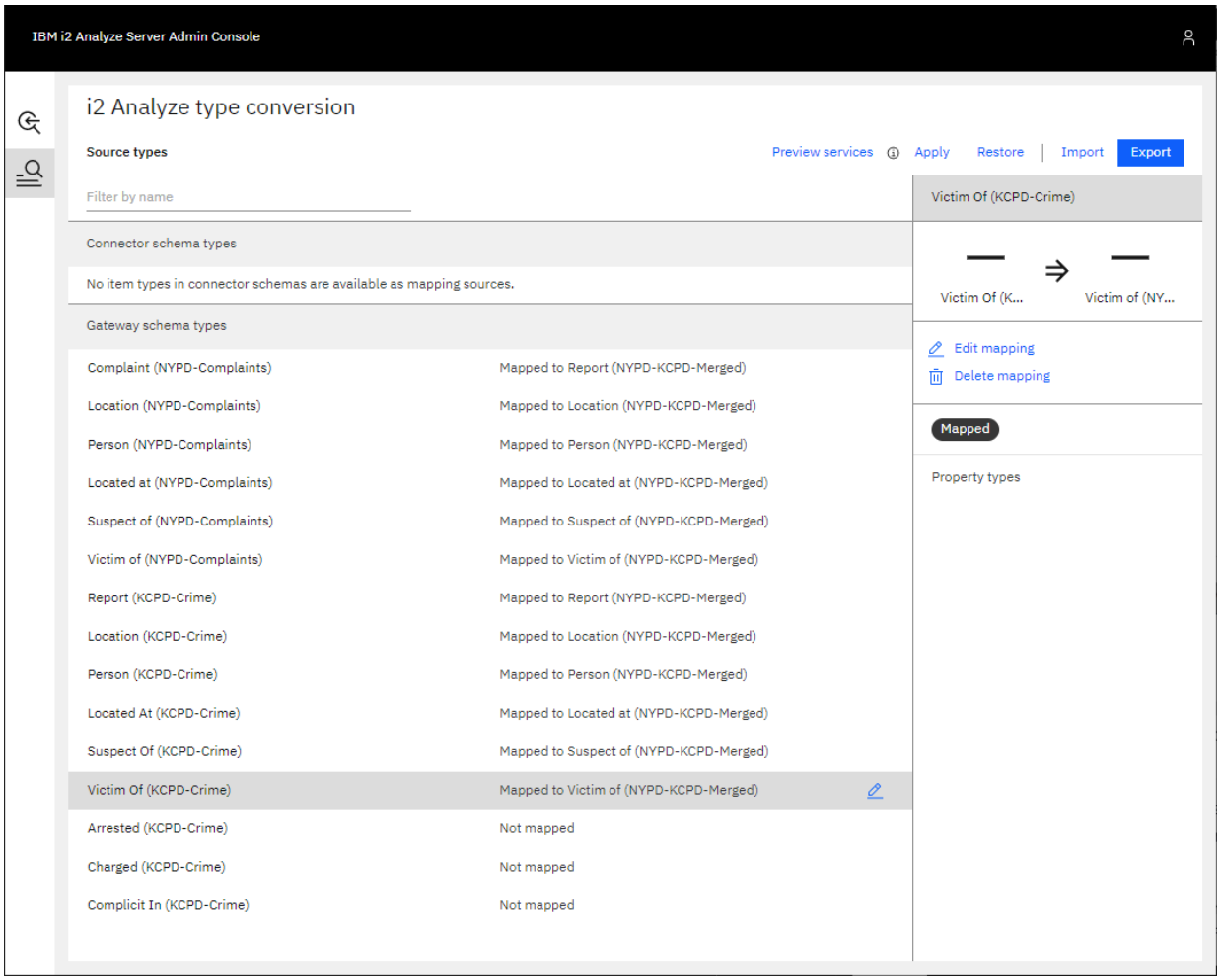
The mapping of Complaint (NYPD-Complaints) to Report (NYPD-KCPD-Merged) might be defined as follows.



Links

The Located At, Suspect Of, and Victim Of link types in both the NYPD-Complaints and KCPD-Crime schemas can be mapped to the corresponding link types in the NYPD-KCPD-Merged schema, which are identical to the link types in NYPD-Complaints.

When all mappings are defined, you should see that all types - except the Complicit In, Arrested, and Charged links from the KCPD-Crime schema - have been mapped to the NYPD-KCPD-Merged schema.



1. Click **Apply** in the top-right. This applies the mappings to the test environment that is available only through the Admin Console. It does not apply the mappings to the live server.
2. Click **Preview services** to open a preview of how the services would behave with the mappings you have configured.
3. Go back and make any changes to the mappings, repeating steps 1 and 2 until you are satisfied with the configuration.

Applying the item type mappings to the i2 Analyze server

To apply the mapping configuration you have created on the i2 Analyze server for all users, see Applying the mapping configuration to the i2 Analyze server.

The result

By creating a new gateway schema and defining item type mappings to it, you have mitigated the problems caused by duplicate or similar item types in the NYPD-Complaints and KCPD-Crime schemas, without being limited to defining mappings in just one direction between them.

Managing i2 Analyze

As the administrator of an i2 Analyze deployment, you have access to tools for ingesting and deleting data from the Information Store; for alerting users when something in the system changes; and for scheduling tasks. You can also call the methods of the i2 Analyze REST API.

Ingesting data into the Information Store

System administrators can add data to the Information Store in bulk by following an ETL (extract, transform, load) process. To update data that was added to the Information Store earlier, they can use the same process to add it again.

Information Store data deletion

i2 Analyze supports several different mechanisms for creating records in the Information Store. It also supports different mechanisms for deleting them individually, selectively, or in bulk.

Retrieving chart metadata

i2 Analyze enables database administrators and third-party extensions to retrieve information about charts that users upload to the Information Store or the Chart Store. Specifically, you can find out which users have uploaded a particular chart, and when that chart was last accessed.

Sending alerts to i2 Analyze users

i2 Analyze enables system administrators and third-party extensions to send alerts to users of the system. These alerts appear in Analyst's Notebook alongside the alerts from Visual Queries that users are familiar with.

Scheduling custom server tasks

i2 Analyze supports the development of custom 'tasks' that the server executes on a user-defined schedule. A task is simply a Java class that implements a specific interface and performs a custom action when invoked.

REST API documentation

i2 Analyze supports a REST API that provides the ability to run text and geospatial searches against the data in the Information Store; to download schemas and server settings; and to perform some server administration functions.

Troubleshooting and support

- i2 Enterprise Insight Analysis support page
- i2 Support

Ingesting data into the Information Store

As the system administrator, you can add data from external sources to the Information Store in bulk by following an ETL (extract, transform, load) process. To keep the Information Store up to date as the contents of the external sources change, you can use the same process to add it again.

To get the most out of this part of the documentation, you need to understand both the i2 Analyze data model and the i2 Analyze schema that defines the data structure of your deployment. Populating the Information Store staging tables also requires you to be familiar with your database management system.

Ingesting data

Information Store data ingestion

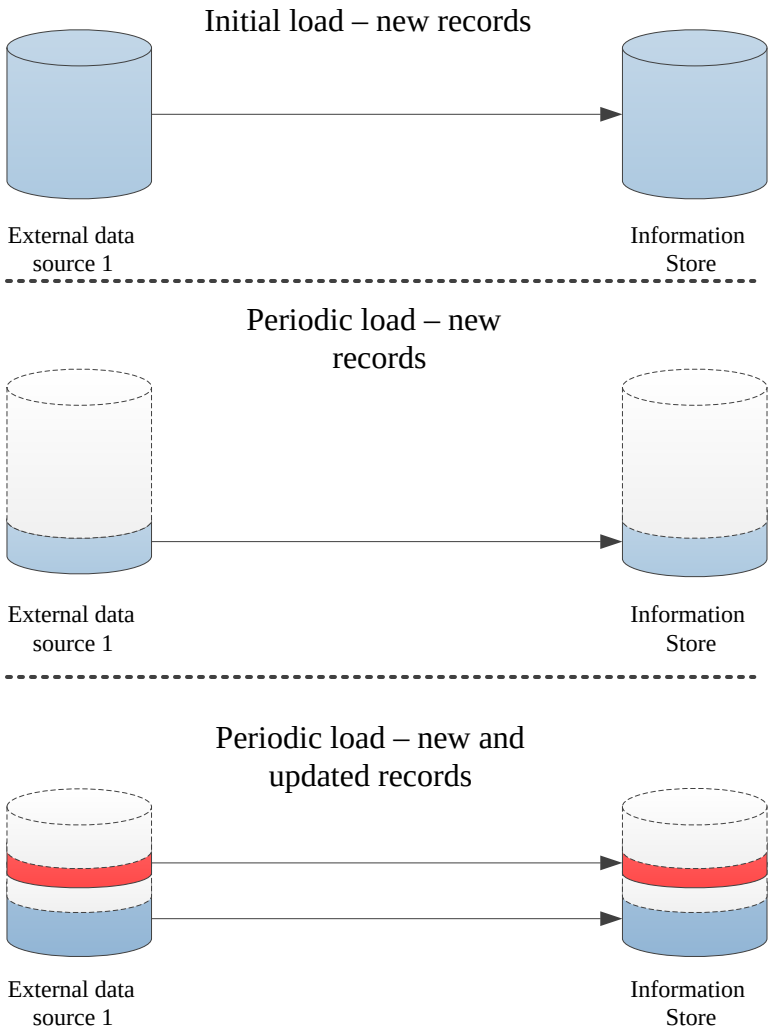
To enable its analytical capabilities, i2® Analyze places requirements on the data records that it processes. i2 Analyze records must conform to the general data model and a deployment-specific i2 Analyze schema, and they need information that might not be present in external sources. Before you can add data from your source to the Information Store, you must transform it to meet the same requirements.

i2 Analyze provides two mechanisms that make it easier to align your data with an i2 Analyze schema and prepare it for ingestion:

- The i2 Analyze deployment toolkit includes a command for creating staging tables that match the structure of the schema, in the same database as the Information Store. If you can load your data into the staging tables successfully, the Information Store can ingest your data from those tables.
- During data ingestion, i2 Analyze uses ingestion mappings that describe how to create records in the Information Store from the data in the staging tables. You can define these mappings in one or more files.

There are two scenarios for ingesting data from an external source. The first time that you load data of a particular item type from an external source, you perform an *initial load*. In an initial load, you are presenting the data to the Information Store for the first time. After you perform the initial load, you might want to perform *periodic loads* that update the Information Store as data is added to the external source or the data is changed. The same staging tables and mapping files can be used in both scenarios.

The following diagram shows the types of data that can be included in each type of load that you might perform. The data in blue is presented to the Information Store for the first time, and the data in red contains updates to data that exists in the Information Store.



The i2 Analyze deployment toolkit provides two *import mode*s that you can use to ingest data:

- **Bulk import mode**

  The *bulk* import mode can be used in the following situations:

  - Initial load, new records, no correlation
  - Periodic load, new records, no correlation

  The bulk import mode is significantly faster for ingesting new, uncorrelated records.
- **Standard import mode**

  The *standard* import mode can be used in the following situations:

  - Initial load, with or without correlation

- Periodic load, new and updated records, with or without correlation

  Standard import mode trades some performance for increased data validation and additional operations during ingestion.

  For more information about correlation, see Overview of correlation.

Because ingestion is completed on a per-item-type basis, you can use the bulk import mode for some item types and use standard mode for others from the external source. You can also use the bulk import mode to complete an initial load, and then use the standard import mode to complete periodic loads to update the data.

Information Store ingestion and the production deployment process

Before you deploy i2 Analyze into production you develop your configuration in a number of environments. The same is true for the ingestion process. You use the same environments to develop the Information Store ingestion process for each of your external data sources.

In the *configuration development environment*, ingest small amounts of test data that is representative of the entity and link types in your existing data sources. By doing this, you can learn how to ingest your data into the Information Store database and ensure that your i2 Analyze schema and security schema are correct for your data. If you discover that the schemas cannot represent your data, you can update your schema files.

You are likely to complete many test ingestions during this development phase. As you understand the process more, you might change how the data is loaded into the staging tables, the data that you use to generate identifiers, or change the ingestion mappings. If you modify the i2 Analyze schema, you must re-create your staging tables to match.

The ingestion mappings that you use are slightly different for each external data source that you ingest data from. It is recommended that you keep any ingestion mapping files in a source control system to track changes and so that you can use them in later development and production environments.

In your *pre-production environment* or *test environment*, use the process that you developed to ingest larger quantities of data and note the time that is taken to ingest the data. You can complete representative initial and periodic loads from each of your external data sources. The time that each ingestion takes allows you to plan when to ingest data in your production environment more accurately.

In your *production environment*, complete your initial loads by using the developed and tested process. According to your schedule, perform your periodic loads on an ongoing basis.

Your deployment architecture can impact the ingestion process in general, and any logic or tools used for transformation of data. For more information, see Understanding the architecture.

The instructions that describe the ingestion process are separated into preparing for ingestion and adding data to the Information Store.

Preparing for ingestion

You must complete three tasks before the Information Store can ingest data from an external source. You must identify exactly which data to load, transform the data to align with the active i2 Analyze schema, and augment the data with extra information that the Information Store requires.

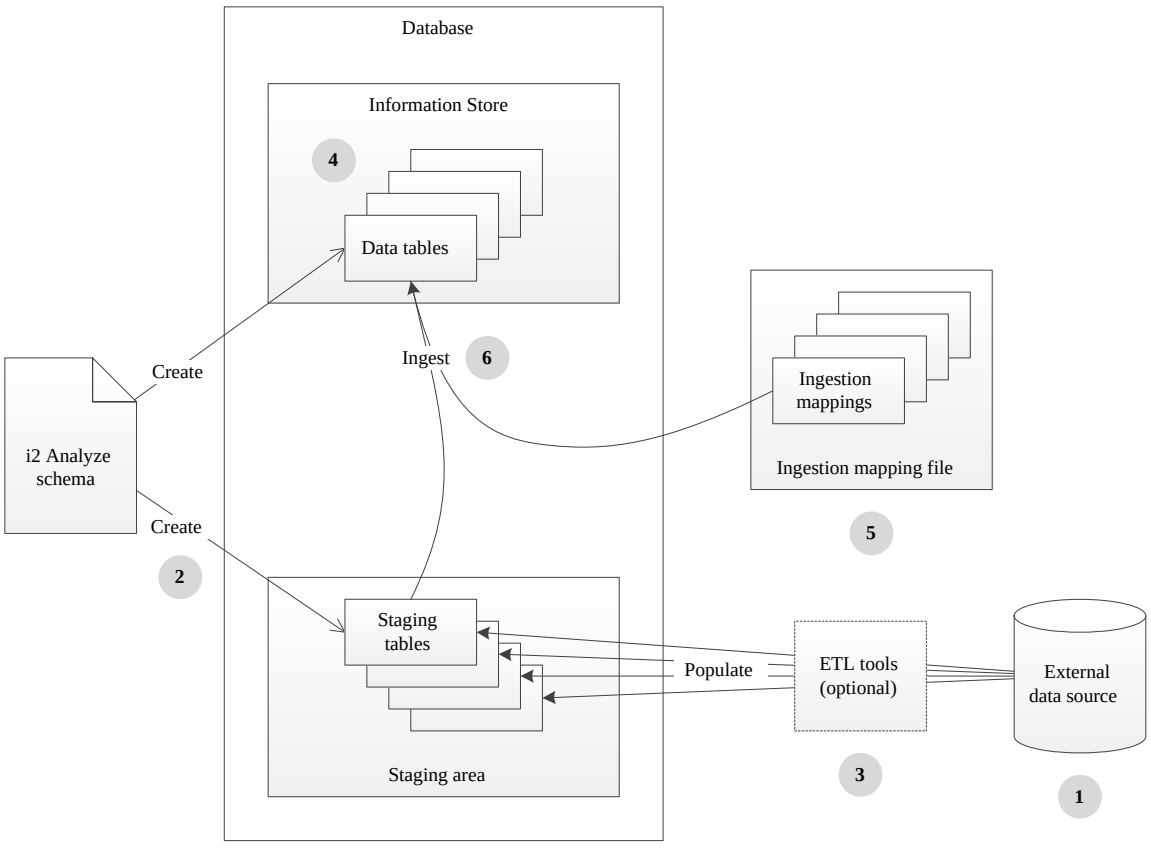
About this task

The only way to add and modify large volumes of data in the i2 Analyze Information Store is to enable and then instruct the Information Store to ingest it. The enablement process involves creating and

populating staging tables for the data, and then supplying the metadata that is crucial to the analytical capabilities of i2 Analyze.

Procedure

You can plan and run the Information Store data ingestion process in a series of discrete steps. This diagram illustrates the approach.



1. Decide which entity types and link types in the active i2 Analyze schema best represent the data that you want the Information Store to ingest.
2. Create staging tables in the database for the types that you identified. Create more than one staging table for some link types.
3. Use external tools, or any other appropriate technique, to transform your data and load the staging tables with the data for ingestion.
4. Add information about your data source to the list of ingestion sources that the Information Store maintains.
5. Write the ingestion mappings that govern the ingestion process and provide additional information that the Information Store requires.
6. Run the ingestion command separately for each of the ingestion mappings that you wrote.

Example

The `examples\data\law-enforcement-data-set-1` and `\signal-intelligence-data-set-1` directories in the deployment toolkit contain files that i2 Analyze uses when you run the `setup -t ingestExampleData` command to populate the Information Store during deployment. These files provide demonstrations of many of the steps in the standard approach to ingestion. The following topics describe those steps in more depth as they detail the Information Store ingestion process.

Identifying the data to be ingested

The detail of how you arrange for the Information Store to ingest your data varies according to how that data is stored in its source. However, the start of the process is always to consider what data you have, and work out how you can shape it to fit the i2 Analyze schema.

About this task

Usually, when you start thinking about adding data from an external source into the Information Store, an i2 Analyze deployment is already in place. That deployment necessarily has an i2 Analyze schema that defines all of the entity types, link types, and property types that data in the system can have. Before you go any further, you must have a clear idea of how your data becomes i2 Analyze entity records and link records in the Information Store.

It is unlikely that the data in your external source has a one-to-one mapping with the entity types and link types in the i2 Analyze schema:

- Probably, your source does not contain data for all the entity types in the schema. As a result, you do not usually need to create a staging table for every possible entity type.
- The schema can define link types that connect several different entity types. In that case, each entity-link-entity type combination for which your source contains data requires a separate staging table.

For example, imagine an i2 Analyze schema that defines the entity types "Person", "Vehicle", and "Account", and the link type "Access to". In this situation, you might decide to create a staging table for each of the entity types. However, the data requires two staging tables for "Access to" links: one for links between people and vehicles, and the other for links between people and accounts.

Procedure

1. Open the schema for the i2 Analyze deployment in Schema Designer.
2. Go through the list of entity types, and determine which of them represent data in your source.
3. Make a note of the identifier of each entity type that represents your data.
4. Repeat steps 2 and 3 for the list of link types. Check the Link Ends tab, and make a note of all the combinations for which your source contains data.

Results

When you complete the steps, you have a list of all the i2 Analyze schema types that your data contains. You also have a list of all the staging tables that you need to create.

Creating the staging tables

The Information Store does not ingest data directly from your data source. Instead, ingestion takes place from staging tables that you create and populate. This abstraction makes it easier for you to align your data with the Information Store, and allows i2 Analyze to validate your mappings before ingestion.

Before you begin

The staging tables that you use to ingest data into the Information Store must conform to a specific structure. For more information about the staging table structure, see Information Store staging tables.

About this task

The simplest approach to Information Store ingestion is to create a staging table for every entity type, and every entity-link-entity type combination, that you identified in your data. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for creating one staging table at a time.

The deployment toolkit command looks like this:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=type_identifier
      -p databaseSchemaName=staging_schema
      -p tableName=staging_table_name
```

While the ETL toolkit command looks like this:

```
createInformationStoreStagingTable
      -stid type_identifier
      -sn staging_schema
      -tn staging_table_name
```

In both cases, *type_identifier* is the identifier of one of the entity types or link types from the i2 Analyze schema that is represented in your data source.

staging_schema is the name of the database schema to contain the staging tables. (If you are using PostgreSQL or SQL Server, the schema must exist before you run the command. If you are using Db2, the command creates the database schema if it does not exist.)

staging_table_name is the name of the staging table itself, which must be unique, and must not exceed 21 characters in length.

Important: Many of the commands that are associated with the ingestion process modify the database that hosts the Information Store. By default, the commands use the database credentials that you specified during deployment in the `credentials.properties` file.
To use different credentials in the deployment toolkit, add `importName` and `importPassword` parameters to the list that you pass to the command. To use different credentials in the ETL toolkit, modify the `DBUsername` and `DBPassword` settings in the `Connection.properties` file.

Procedure

1. If you are using the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory. If you are using the ETL toolkit, navigate to the `etltoolkit` directory.
2. For each entity type or link type that you identified for ingestion, run the `createInformationStoreStagingTable` command.
   For example:

   ```
   setup -t createInformationStoreStagingTable
         -p schemaTypeId=ET5 -p databaseSchemaName=IS_Staging
         -p tableName=E_Person
   ```

By convention, you create all of the staging tables for the same source in the same database schema, which has the name `IS_Staging` in this example. It is also conventional to name the staging table itself similarly to the display name of the entity type or link type to which the table corresponds. In this case, the staging table is for the Person entity type.

Note: When the i2 Analyze schema allows the same link type between several different entity types, create separate staging tables for each combination:

```
setup -t createInformationStoreStagingTable
      -p schemaTypeId=LAC1 -p databaseSchemaName=IS_Staging
      -p tableName=L_Access_To_Per_Acc

setup -t createInformationStoreStagingTable
      -p schemaTypeId=LAC1 -p databaseSchemaName=IS_Staging
      -p tableName=L_Access_To_Per_Veh
```

This example illustrates an Access To link type (with identifier `LAC1`) that can make connections from Person entities to Account entities, or from Person entities to Vehicle entities. The commands create staging tables with different names based on the same link type.

Results

At the end of this procedure, you have a set of staging tables that are ready to receive your data before ingestion takes place. The next task is to make your data ready to populate the staging tables.

Preparing the external data

The staging tables that you create during the ingestion process have data structures that are similar to, but simpler than, the Information Store data tables. Whatever your source is, you must find a way to shape the data that it contains into a form that is compatible with the staging tables.

About this task

After you create the staging tables, you can view them in Microsoft SQL Server Management Studio (or similar software) to see the definitions of their columns. You must make your data matches these definitions before you can go on to populate the staging tables.

Important: The Information Store and client applications place limits on the ranges of values that properties with different logical types can contain. If you attempt to use values outside these ranges, failures can occur during ingestion or when that data is presented to users. Before you ingest your data, you must ensure that it conforms to the ranges specified in Information Store property value ranges.

Procedure

Because all data sources and many i2 Analyze schemas are different, there is no single procedure that you can follow to prepare your data for ingestion. However, there are a number of common considerations.

- Each staging table can contain data that maps to only one entity type or link type. If your source data has rows or records that contain data for more than one of the types that you identified, then you must separate them during preparation or population.

  For data in a relational source, this preparation might mean creating views on the original tables. If the data is in CSV files, then you might need to wait until you populate the staging tables to change its shape in this way.

- The Information Store does not support storing properties with multiple values in the same i2 Analyze record. The records that you create must contain values for a maximum of one property with each permitted property type.
- If you are dealing with date and time data, that data must meet extra requirements before the Information Store can ingest it. To retain information unambiguously, the staging tables use four columns to represent date and time data.

  Even if you know that your date and time data was recorded in Coordinated Universal Time, you must make that fact explicit in the data to be ingested. For example, if your source contains information about an event that started at 9 AM on October 6 2002, then the values you need to prepare are:

  - `2002-10-06 09:00:00` (the data and time originally entered)
  - `UTC` (the time zone)
  - `0` (Daylight Saving Time is not in effect)
  - `2002-10-06 09:00:00` (the date and time in Coordinated Universal Time)
- The `source_id`, `origin_id_type`, `origin_id_keys` columns of the staging table are used to store values that reference the data in its original source and can be used to make up the origin identifier of the resulting record.

  Note: If the staging table definition was for a link type, it would also contain `from_` and `to_` variations of each of the columns.
- If your external source is a relational database, you might find that the only data for some links is the presence of a foreign key relationship between two tables. In that case, you must synthesize a reproducible reference for the link from the other data that you have available.

  For example, you might be able to create a unique reference for a link by combining the identifiers of the entity records at its ends.
- All staging tables contain a `source_last_updated` column that you can use to store information about when the data to be ingested was modified in its source.
- All staging tables contain columns for each of the access dimensions that the security schema defines. If your external source includes security information, then you can map that information to the security schema of your target deployment, and populate the staging table columns accordingly.

  Alternatively, you can leave the security columns blank, and provide security dimension values on a mapping- or source-wide basis later in the ingestion process.
- All staging tables contain `correlation_id_type` and `correlation_id_key` columns. To correlate data that is ingested into the Information Store, use these columns to store the values that comprise the correlation identifier for each row of data. If you do not want to use correlation, leave the columns blank.

  If you specify values for a correlation identifier, then also specify a value for the `source_last_updated` column, which is used during the correlation process.

  For more information about correlation, correlation identifiers, and the impact of the `source_last_updated` value, see Overview of correlation.
- The columns named `source_ref_source_type`, `source_ref_source_location`, and `source_ref_source_image_url` are used to populate the source reference that is generated when the data is ingested.

  For more information about implementing source references in your deployment, see Configuring source references.

- The staging tables for link types contain a column for the direction of the link.

  The Information Store considers links to go "from" one entity "to" another. The direction of a link can be `WITH` or `AGAINST` that flow, or it can run in `BOTH` directions, or `NONE`.

  - If your link data includes direction information, then you can add it to the staging table during the population process, and then refer to it from the mapping file.
  - If your link data does not include direction information, then you can specify a value in the mapping file directly.

  By default, if you have no direction information and you do nothing in the mapping file, the Information Store sets the direction of an ingested link to `NONE`.

Example

The `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit contains a set of CSV files that were exported from a relational database.

In files like `event.csv`, you can see date and time data that meets the requirements of the staging tables. You can also see multiple files for "Access to" links, and how some staged link rows contain little more than a set of identifiers.

Populating the staging tables

The i2 Analyze deployment toolkit and the ETL toolkit create the staging tables for data ingestion in the same database as the Information Store data tables. After you prepare your data, but before you can instruct the Information Store to ingest it, you must populate the staging tables.

About this task

The approach that you take to populate the staging tables depends on the database management system that you are using, the form that your source data is in, and the tools that you have available.

PostgreSQL provides the `\COPY` (and `COPY`) and `INSERT` commands:

- If your data is in comma-separated value (CSV) files, you can use either the `psql` client tool's `\COPY` command, or the SQL `COPY` command.
- If your data is in the tables or views of another database, you can use the SQL `INSERT` command.

SQL Server provides the bulk insert and insert utilities:

- If your data is in comma-separated value (CSV) files, then you can use the `BULK INSERT` command.

  Note: To use the `BULK INSERT` command, the user that you run the command as must be a member of the `bulkadmin` server role.
- If your data is in the tables or views of another database, then you can use the `INSERT` command.
- You can use SQL Server Integration Services as a tool to extract and transform data from various sources, and then load it into the staging tables.

Db2 provides the ingest, import, and load utilities:

- If your data is in comma-separated value (CSV) files, then you can use the `IMPORT` or `INGEST` commands.
- If your data is in the tables or views of another database, then you can use the `IMPORT`, `INGEST`, or `LOAD` commands.

Alternatively, regardless of your database management system, you can use IBM InfoSphere DataStage as a tool for transforming your data and loading it into the staging tables. You can specify the database schema that contains the staging tables as the target location for the ETL output.

Example

The subdirectories of the `examples\data` directory in the deployment toolkit all contain a `postgres`, a `sqlserver`, and a `db2` directory.

If you are using PostgreSQL, inspect the `LoadCSVDataCommands.sql` file in the `postgres` directory. In each case, this file is an SQL script that populates the example staging tables from the prepared CSV files. The script calls the `\COPY` command repeatedly to do its work.

If you are using SQL Server, inspect the `LoadCSVDataCommands.sql` file in the `sqlserver` directory. In each case, this file is an SQL script that populates the example tables from the prepared CSV files. The script calls the `BULK INSERT` command repeatedly to do its work. The `BULK INSERT` command uses `.fmt` format files, which are also in the `sqlserver` directory, to instruct SQL Server how to process the CSV files into the staging tables. For more information about format files, see Use Non-XML format files.

If you are using Db2, inspect the `LoadCSVDataCommands.db2` file in the `db2` directory. In each case, this file is a Db2 script that populates the example staging tables from the prepared CSV files. The script calls the `IMPORT` command repeatedly to do its work. In most instances, the command just takes data from columns in a CSV file and adds it to a staging table in a Db2 database schema.

Defining an ingestion source

The Information Store keeps a list of all the sources from which it has ingested data. Before it can ingest data, you must tell the Information Store about your source. In the ingestion mapping file, you then specify the data source name in the mapping definition for each entity type and link type.

About this task

The i2 Analyze deployment toolkit and the ETL toolkit both have a command for adding information about an ingestion source to the Information Store.

The deployment toolkit command looks like this:

```
setup -t addInformationStoreIngestionSource
      -p ingestionSourceName=src_name
      -p ingestionSourceDescription=src_display_name
```

While the ETL toolkit command looks like this:

```
addInformationStoreIngestionSource
      -n src_name
      -d src_display_name
```

In both cases, `src_name` is a unique name for the ingestion source, which also appears in the mapping file. `src_display_name` is a friendlier name for the ingestion source that might appear in the user interface of applications that display records from the Information Store.

Important: The value that you provide for `src_name` must be 30 characters or fewer in length. Also, do not use the word `ANALYST` as the name of your ingestion source. That name is reserved for records that analysts create in the Information Store through a user interface.

Procedure

1. If you are using the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory. If you are using the ETL toolkit, navigate to the `etltoolkit` directory.
2. Run the `addInformationStoreIngestionSource` command, specifying the short and display names of your ingestion source.

   For example:

   ```
   setup -t addInformationStoreIngestionSource
         -p ingestionSourceName=EXAMPLE
         -p ingestionSourceDescription="Example data source"
   ```

   If the Information Store already contains information about an ingestion source with the name `EXAMPLE`, this command has no effect.

Results

After you complete this task, you have performed all the necessary actions, and gathered all the necessary information, to be able to write ingestion mapping files. The next task is to create the ingestion mapping file for your ingestion source.

Creating an ingestion mapping file

The mappings in an ingestion mapping file define how rows in staging tables become i2 Analyze records in the Information Store during the ingestion process. Each mapping that you write describes how to construct the origin identifiers for data of a particular type, and specifies the security dimension values that apply to records.

Before you begin

Before you create your ingestion mappings, review the information about origin identifiers and security dimension values in i2 Analyze.

About this task

The Information Store ingestion mechanism makes it possible for you to develop and extend your ingestion mappings over time. You can test your approach to ingestion by writing and using a single (entity type) mapping, and then adding more entity type and link type mappings later. You can put all your mappings in one file, or put each mapping in a separate file, or anything between those two extremes.

Procedure

If you populated the staging tables successfully, then writing ingestion mappings can be straightforward. Eventually, you need a mapping for each staging table that you created, but you can approach the problem one mapping at a time.

1. Choose a populated staging table for an entity type that has links to other entity records in the data model.
2. Create an ingestion mapping file that contains an ingestion mapping for the staging table that you chose in step 1. If you prefer to start from an existing file, look at `mapping.xml` in the `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit.
3. Run the ingestion command to validate the mapping.

   If you are unhappy with the outcome, edit the ingestion mapping and run the command again.

4. Repeat all of the preceding steps for all the other staging tables that you populated.

Example

The `examples\data\law-enforcement-data-set-1` directory of the deployment toolkit contains an ingestion mapping file named `mapping.xml`. This file contains ingestion mappings for all the staging tables that the ingestion example creates. You can use `mapping.xml` as the basis for the ingestion mappings that you need for your data.

Validating the ingestion configuration

After you create and populate your staging tables and write your ingestion mappings, the final part of the process is to run the command to validate your work.

About this task

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for validating the structure of your staging tables and a particular ingestion mapping in a particular mapping file.

The data within the staging tables is not validated against the i2 Analyze schema or allowed value ranges. For more information about the value ranges, see Information Store property value ranges.

Procedure

The procedure for instructing the Information Store to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. Choose an entity staging table that you populated with data and provided with an ingestion mapping.
2. Run the `ingestInformationStoreRecords` command in `VALIDATE` mode.
   For example:

   ```
   setup -t ingestInformationStoreRecords -p importMappingsFile=mapping.xml
         -p importMappingId=Person -p importMode=VALIDATE
   ```

   When you specify `importMode` and set it to `VALIDATE`, the command checks the validity of the specified mapping, but no ingestion takes place. For more information about the running the command and any arguments, see The ingestInformationStoreRecords task.
   The output to the console indicates whether the mapping you identified is valid, provides guidance when it is not valid, and gives a full list of column mappings. The command sends the same information to a log file that you can find at `toolkit\configuration\logs\importer\i2_Importer.log`.
3. Complete this validation step for each staging table and ingestion mapping that you plan to use. The ingestion process for links verifies that the entities at each end of the link are already ingested. If it fails to find them, the process fails. When you are developing your ingestion process, ingest a small amount of entity data before you validate your links.

What to do next

Correct any problems in the ingestion mappings file (or any ingestion properties file that you specified) before you proceed to Adding data to the Information Store.

Adding data to the Information Store

After you populate the staging tables and write ingestion mappings, you can use toolkit commands to instruct the Information Store to ingest or update the records that represent external data. The Information Store keeps a log of all such instructions that you can review to determine the success or failure of each one.

About this task

The commands in the i2 Analyze deployment and ETL toolkits make it possible for you to create, update, and delete records in the Information Store. All three operation types are controlled by the data in the staging tables and the mappings in the ingestion mapping files.

After any operation that uses toolkit commands to change the contents of the Information Store, you can examine ingestion reports to determine how successful the operation was.

As described in Information Store data ingestion, there are two different import modes that you can use to ingest your data. Before you run the ingestion commands, ensure that you use the correct import mode for the data that you want to ingest. Remember that this might differ depending on the item type that you are ingesting or the ingestion mapping that you are using.

Using bulk import mode

Bulk import mode can be used for improved ingestion performance when you are populating the Information Store with new records as part of an initial or periodic load.

Before you begin

- Before you run the ingestion commands, ensure that you complete all of the tasks in Preparing for ingestion.
- For Db2, the mechanism that bulk import mode uses to load the data from the staging tables to the Information Store can require changes to the database configuration. Before you ingest data, ensure that your database is configured correctly. For more information, see Database configuration for IBM Db2.
- You can drop and re-create the database indexes during the ingestion process, which can reduce the total time that it takes to ingest data. For more information about when it is beneficial to drop and re-create the database indexes, see Database index management

About this task

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for ingesting the data that is associated with a particular ingestion mapping in a particular mapping file. For more information about the running the command and any arguments, see The `ingestInformationStoreRecords` task.

When you use bulk import mode, take the following points into consideration:

- For best performance, ensure that there are no analysts using the system during the ingestion process.
- Do not include correlation identifiers in the data that you ingest by using bulk import mode. If any correlation identifiers are included, the ingestion fails.
- You cannot run concurrent `ingestInformationStoreRecords` commands when using bulk import mode.

- Source references are not ingested when using bulk import mode. To ingest source references, you can use the standard import mode afterwards to update the ingested records with their source references.

Procedure

After you stage your data and create your configuration files, you can instruct the Information Store to ingest your data by using bulk import mode. The procedure to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. If you have decided to drop the database indexes, do so now.
2. Use the following command to ingest your data.

   ```
   setup -t ingestInformationStoreRecords
   -p importMappingsFile=ingestion_mapping_file
   -p importMappingId=ingestion_mapping_id
   -p importLabel=ingestion_label
   -p importConfigFile=ingestion_settings_file
   -p importMode=BULK
   ```

   For more information about the running the command and any arguments, see The `ingestInformationStoreRecords` task.
3. If any errors occur, refer to Troubleshooting the ingestion process.
4. If you need to re-create the database indexes, do so now.

Database configuration for IBM Db2

When you use bulk or bulk delete import modes, the way that the data is inserted into or removed from the Information Store from the staging table requires more transaction log space and database locks available than the standard modes.

Transaction log size

For entity operations, allocate at least 1.5 GB of transaction log space.

For link operations, allocate at least 22 GB of transaction log space.

For more information about changing the Db2 log file size, see:

- logfilsiz - Size of log files configuration parameter,
- logprimary - Number of primary log files configuration parameter, and
- logsecond - Number of secondary log files configuration parameter.

Lock list size

For entity operations, the formula for calculating the required lock list size is:

```
1200000 * (lock_record_size / 4000)
```

If the lock record size for your deployment is 256 bytes, you need a lock list size of 75,000.

For link operations, the formula for calculating the required lock list size is:

```
3600000 * (lock_record_size / 4000)
```

If the lock record size for your deployment is 256 bytes, you need a lock list size of 225,000. Link ingestion requires an increased lock list size because more tables in the Information Store are written to.

For information about lock lists, see locklist - Maximum storage for lock list configuration parameter.

Database index management

It can be beneficial for performance to drop and re-create indexes in the Information Store database during bulk mode ingestion. The situations when dropping and creating the database indexes might help change over the lifetime of a deployment as your Information Store contains more data.

Decide when to drop and create indexes

The situations where you might want to drop and re-create the database indexes include:

- Completing an initial load for an item type or ingesting data into an empty Information Store
- When ingesting large amounts of data (for example, more than 20 million rows in the staging table)

The situations where you do not want to drop and re-create the database indexes include:

- When the Information Store contains a large amount of data for the item type you are ingesting, the time that it takes to re-create the indexes can make the total ingestion time longer. If you are ingesting into an Information Store that already contains about 1 billion records, do not drop and re-create the database indexes.
- If you are ingesting links between entities of the same type and some of those entities might be correlated, do not drop and re-create the database indexes.

By default, the indexes are kept in place and not dropped during data ingestion.

To help determine what is best for your data, you can test some ingestions that drop and re-create indexes and some that don't. When you complete the test ingestions, record the time that it takes to complete. As the amount of data in the Information Store increases, the time it takes to create the indexes also increases.

Dropping and re-creating indexes during a bulk import ingestion process

If you decide to drop and re-create the database indexes during large ingestions that use multiple staging tables across both entity and link types, the high-level process consists of the following steps:

1. Stop Liberty and Solr
2. Drop the database indexes for the entity types that you are ingesting data for
3. Ingest all entity data
4. Create the database indexes for the entity types that you ingested data for
5. Drop the database indexes for the link types that you are ingesting data for
6. Ingest all link data
7. Create the database indexes for the link types that you ingested data for
8. Start Liberty and Solr

You can use two methods to drop and create indexes:

- Use the i2 Analyze deployment toolkit to generate scripts that you run against the database manually.
- Use the import configuration properties file to specify whether the indexes must be dropped or created during a bulk import mode ingestion.

If you are ingesting data for multiple link types or you want to review the database scripts and run them manually, it is best to use the generated scripts to drop and re-create the database indexes. If you do not want to (or cannot) run scripts against the Information Store database, you can use the import configuration file and allow the ingestion process to drop and re-create the database indexes.

If you are completing an ingestion that spans multiple ingestion periods where the system is in use between ingestion periods, ensure that all of the indexes are created at the end of an ingestion period and that you start Liberty and Solr before analysts use the system.

For information about creating the import configuration file, see References and system properties.

Using the generated scripts method

1. In your import configuration file, set both the `dropIndexes` and `createIndexes` settings to `FALSE`.

   For example:

   ```
   dropIndexes=FALSE
   createIndexes=FALSE
   ```

2. Generate the *create* and *drop* index scripts for each item type that you are ingesting data for.

   For example, to generate the scripts for the item type with identifier `ET5`, run the following commands:

   ```
   setup -t generateInformationStoreIndexCreationScripts -p schemaTypeId=ET5
   setup -t generateInformationStoreIndexDropScripts -p schemaTypeId=ET5
   ```

   The scripts are output in the `toolkit\scripts\database\db2\InfoStore\generated` directory, in the `createIndexes` and `dropIndexes` directories.

3. Stop Liberty and Solr.

   On the Liberty server, run:

   ```
   setup -t stopLiberty
   ```

   On each Solr server, run:

   ```
   setup -t stopSolrNodes --hostname solr.host-name
   ```

   Where `solr.host-name` is the host name of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

4. Run the scripts that you generated in Step 2 to drop the indexes for each entity type that you plan to ingest data for.
5. Complete the process to ingest the entity data using the bulk import mode. For more information, see Using bulk import mode.
6. Run the scripts that you generated in Step 2 to create the indexes for each entity type that you ingested.
7. Run the scripts that you generated in Step 2 to drop the indexes for each link type that you plan to ingest data for.
8. Complete the process to ingest the link data using the bulk import mode. For more information, see Using bulk import mode.

   Only re-create the database indexes after you ingest the link data for every link type that you plan to ingest.
9. Run the scripts that you generated in Step 2 to create the indexes for each link type that you ingested.
10. Start Liberty and Solr.

    On each Solr server, run:

```
setup -t startSolrNodes --hostname solr.host-name
```

Where `solr.host-name` is the host name of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.

On the Liberty server, run:

```
setup -t startLiberty
```

Using the import configuration properties file method

When you use the import configuration settings to drop and re-create the database indexes, the toolkit creates and drops the indexes for you as part of the ingestion process instead of running scripts against the database manually. However, you must modify the import configuration file a number of times throughout the ingestion of multiple item types. You must still stop Liberty and Solr before you run the ingestion command, and start them again after the indexes are created.

If all of the data for a single item type is ingested with a single ingestion command, in your import configuration file set the `dropIndexes` and `createIndexes` settings as follows:

```
dropIndexes=TRUE
createIndexes=TRUE
```

If the data for a single item type must be ingested by using multiple ingestion commands, you need to modify the import configuration file before the first ingestion command, for the intermediate commands, and before the final command for each item type. For example, if your data for a single entity type is in more than one staging table:

1. The first time you call the ingestion command, set the `dropIndexes` and `createIndexes` settings as follows:

   ```
   dropIndexes=TRUE
   createIndexes=FALSE
   ```

2. For the intermediate times that you call the ingestion command, set the `dropIndexes` and `createIndexes` settings as follows:

   ```
   dropIndexes=FALSE
   createIndexes=FALSE
   ```

3. The final time you call the ingestion command, set the `dropIndexes` and `createIndexes` settings as follows:

   ```
   dropIndexes=FALSE
   createIndexes=TRUE
   ```

After you ingest all the entity data, repeat the process for the link data.

Using standard import mode

After you create and populate your staging tables, write your ingestion mappings, and validate your work, instruct the Information Store to ingest your data.

Before you begin

If you are using correlation, you must ensure that the application server is started before you run the ingestion commands.

About this task

When you instruct the Information Store to ingest the data that you loaded into the staging tables, you do it one ingestion mapping (and one staging table) at a time. The i2 Analyze deployment toolkit and the ETL toolkit both have a command for ingesting the data that is associated with a particular ingestion mapping in a particular mapping file. For more information about the running the command and any arguments, see The `ingestInformationStoreRecords` task.

Procedure

The procedure for instructing the Information Store to ingest your data is similar to many others in this process. You start with one type or one staging table, and build from there.

1. Choose an entity staging table that you populated with data and provided with an ingestion mapping.
2. Run the `ingestInformationStoreRecords` command in `STANDARD` mode to instruct the Information Store to ingest your data. For example:

   ```
   setup -t ingestInformationStoreRecords
         -p importMappingsFile=ingestion_mapping_file
         -p importMappingId=ingestion_mapping_id
         -p importLabel=ingestion_label
         -p importConfigFile=ingestion_settings_file
         -p importMode=STANDARD
   ```

   For more information about the running the command and any arguments, see The `ingestInformationStoreRecords` task.

   Note: You can improve the performance of entity ingestion by running `ingestInformationStoreRecords` for different entity types at the same time when you use the standard import mode. It is recommended that you use the ETL toolkit to run concurrent `ingestInformationStoreRecords` commands. *Do not* attempt to run the command for data of the same type at the same time.

   Due to the increased number of operations and comparisons that are required, using correlation can make the ingestion process take longer.
3. Repeat steps 1 and 2 for the other staging table and ingestion mapping combinations that you created. Take care to run the command for entity types before you run it for link types.

   The ingestion process for links verifies that the entities at each end of the link are already ingested. If it fails to find them, the process fails.

Results

At the end of this procedure, all the external data that you populated the staging tables with is in the Information Store. To add or update records, you can repopulate the staging tables and rerun the `ingestInformationStoreRecords` command.

Updating the Information Store for changed data

The data that the Information Store ingests is fixed at the moment of ingestion, and changes to the data in its source do not automatically update the Information Store. However, you can update the

Information Store to reflect changes in an external source by running through the ingestion process again.

About this task

For most changes to the data in an external source, it is likely that you can reuse the work that you did to enable initial ingestion. If the changes to an external source are not significant enough to affect your method for generating reproducible origin identifiers, repeat ingestion follows the same process as initial ingestion.

Procedure

1. Examine the new or changed data in the external source, and your ingestion mappings. Confirm that your configuration still generates origin identifiers that the Information Store can compare with their equivalents in existing ingested data.
2. Delete the contents of each staging table that you know to be affected by changes to the external data.
3. Populate the affected staging tables with the latest data from your external source.
4. Run the ingestion command specifying the standard import mode for each ingestion mapping that refers to an affected staging table, taking care to process entity data before link data, as usual.

   The Information Store uses the origin identifier of each row that it attempts to ingest to determine whether the data is new:

   - If the origin identifier does not match the origin identifier of any data that is already in the Information Store, then the data is new to the Information Store. It is ingested in the usual way.
   - If the origin identifier does match the origin identifier of any data that is already in the Information Store, then the staging table contains updated information. The Information Store clears its existing data and refills it with the new data.

     Note: If the correlation identifier changed, additional merge and unmerge operations might occur.

Results

After you follow this procedure, the Information Store contains new data that was added to an external source since the last ingestion. It also contains updated data that was changed in an external source since the last ingestion.

Updating the Information Store for deleted data

The data that the Information Store ingests is fixed at the moment of ingestion, and removal of the data in the external source does not automatically delete it from the Information Store. However, you can update the Information Store to reflect the deletion of data in the external source by using staging tables and the deployment toolkit.

Before you begin

When data changes in its original source, you can use the same pipeline that you used for initial ingestion to update the records in the Information Store. If data is deleted from its source, you can use the staging tables and the deployment toolkit to reflect that fact in the Information Store as well.

A single i2 Analyze record can represent data from multiple sources, which results in a record that contains multiple pieces of provenance. As a consequence, responding to source data deletion does not necessarily mean deleting records from the Information Store. When you use the toolkit to reflect

deleted source data, the effect is to remove the provenance associated with that data. If the process removes a record's only provenance, the record is deleted. If not, the record remains in the Information Store.

Note: To delete records from the Information Store explicitly, use the deletion-by-rule approach. You can write conditions to determine which records are deleted. For more information about deleting records in this way, see Deleting records by rule.

About this task

The commands to update the Information Store for deleted data use the same mapping file and staging tables as the commands for ingesting data, and you call them in a similar way. However, the only information that *must* be in the staging table is what the mapping file requires to generate the origin identifiers of the data that is no longer in the external source.

When you run the commands to update the Information Store for deleted data, the rules that apply differ from the rules for adding and updating data:

- Links do not have to be processed before entities, or vice versa.
- Links can be processed without specifying the origin identifiers of their ends.
- Deleting a piece of provenance from an entity record also deletes all the link provenance that is connected to it.
- The process silently ignores any origin identifiers that are not in the Information Store.

Because this process might cause significant numbers of i2 Analyze records to be deleted, two commands are provided. The first command previews the effect of running the second command before you commit to doing so. In the deployment toolkit, the two commands have different names but the same syntax:

```
setup -t previewDeleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
```

```
setup -t deleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p logConnectedLinks
      -p importMode=BULK_DELETE
```

In the ETL toolkit, you reuse the `ingestInformationStoreRecords` command. For more information about running the command from the ETL toolkit, see the *ETL toolkit* section of The `ingestInformationStoreRecords` task.

For more information about the running the commands and any arguments, see The `previewDeleteProvenance` and `deleteProvenance` tasks.

Bulk delete mode can be used for improved performance when you are removing provenance from the Information Store that does not contribute to correlated records. If you try to delete any provenance that contributes to correlated records, that provenance is not removed from the Information Store and is recorded in a table in the `IS_Public` database schema. The table name is displayed in the console when the delete process finishes. For example, `IS_Public.D22200707130930400326011ET5`. Before you use bulk delete mode, ensure that your database is configured correctly. For more information, see Database configuration for IBM Db2.

Procedure

The procedure for updating the Information Store in this way starts with a staging table that contains information about the data that you no longer want to represent in the Information Store.

1. Ensure that the application server that hosts i2 Analyze is running.
2. Run the `previewDeleteProvenance` command to discover what the effect of running `deleteProvenance` is. For example:

   ```
   setup -t previewDeleteProvenance -p importMappingsFile=mapping.xml
         -p importMappingId=Person
   ```

   The output to the console window describes the outcome of a delete operation with these settings. High counts or a long list of types might indicate that the operation is going to delete more records than you expected. Previewing the delete operation does not create an entry in the `Ingestion_Deletion_Reports` view, the output is displayed in the console.

   ```
   >INFO [DeleteLogger] - Delete preview requested at 2017.12.08 11:05:32
   >INFO [DeleteLogger] - Item type: Person
   >INFO [DeleteLogger] - Number of 'Person' provenance pieces to be deleted:
    324
   >INFO [DeleteLogger] - Number of 'Person' i2 Analyze records to be
    deleted: 320
   >INFO [DeleteLogger] - Number of 'Access To' provenance pieces to be
    deleted: 187
   >INFO [DeleteLogger] - Number of 'Access To' i2 Analyze records to be
    deleted: 187
   >INFO [DeleteLogger] - Number of 'Associate' provenance pieces to be
    deleted: 27
   >INFO [DeleteLogger] - Number of 'Associate' i2 Analyze records to be
    deleted: 27
   >INFO [DeleteLogger] - Number of 'Employment' provenance pieces to be
    deleted: 54
   >INFO [DeleteLogger] - Number of 'Employment' i2 Analyze records to be
    deleted: 54
   >INFO [DeleteLogger] - Number of 'Involved In' provenance pieces to be
    deleted: 33
   >INFO [DeleteLogger] - Number of 'Involved In' i2 Analyze records to be
    deleted: 33
   >INFO [DeleteLogger] - Duration: 1 s
   ```

   Note: When you run the command for entity records, the output can exaggerate the impact of the operation. If the staging table identifies the entities at both ends of a link, the preview might count the link record twice in its report.
3. Correct any reported problems, and verify that the statistics are in line with your expectations for the operation. If they are not, change the contents of the staging table, and run the preview command again.
4. Run the `deleteProvenance` command with the same parameters to update the Information Store. For example:

   ```
   setup -t deleteProvenance -p importMappingsFile=mapping.xml
         -p importMappingId=Person -p importLabel=DeletePeople
         -p logConnectedLinks
   ```

   Note: Do not run multiple `deleteProvenance` commands at the same time, or while data is being ingested into the Information Store.
5. Repeat the steps for the types of any other records that you want to process.

Results

At the end of this procedure, the Information Store no longer contains the provenance (or any connected link provenance) for the data that you identified though the mapping files and staging tables. Any records that lose all of their provenance, and any connected link records, are deleted as a result. Deleting data is permanent, and the only way to restore it to the Information Store is to add it again through the `ingestInformationStoreRecords` command.

Monitoring the search index

After an ingestion operation, i2 Analyze updates its search index to reflect the changes to the Information Store. You can monitor the progress of this process, and intervene if the server encounters problems.

Successful ingestion doesn't always mean that indexing will also be successful. There are some circumstances under which malformed data can make it fail. If indexing *does* fail, however, you don't have to re-ingest all the data. Rather, you can locate the batch that caused the problem and correct the issue there.

i2 Analyze provides three complementary ways of monitoring the state of the search index:

- **REST endpoint**

  `GET /api/v1/admin/indexes/status` provides information about the health of all the indexes that i2 Analyze maintains. The name of the search index is `main_index`, and when it's healthy the response from the endpoint includes the following output:

  ```
  {
    "status": {
      ...
      "main": [
        {
          "name": "main_index",
          "populationPaused": false,
          "populationProgress": 100,
          "state": "LIVE",
          "statusMessage": "The Solr collection is healthy. The minimum
   replication factor can be achieved.",
          "hasFailedBatches": false
        }
      ],
      ...
    }
  }
  ```

- **Indexing log file**

  i2 Analyze outputs messages about the search index to the log file at `wlp\usr\servers\opal-server\logs\opal-services\main_index\i2_Indexer.log`.

- **Database view**

  The Information Store database includes a public view named `IS_Public.Indexing_Failed_Batches` that contains information about problems with the indexing process. It can look like this:

batch_id	time_stamp	schema_type_id	error_message
50	2023-07-25 14:56:09.919000	ET1	...
52	2023-07-25 14:56:20.147000	ET5	...

Detecting problems

To monitor the search index for problems that occur after data ingestion - and to find the cause of any such problems - you can use these tools together:

1. Poll the `indexes/status` endpoint periodically. In particular, watch out for the main index's `populationProgress` falling below 100 for an extended time, and for `hasFailedBatches` being `true`.
2. Alternatively or additionally, check the `i2_Indexer.log` file for any messages that begin with "Failed to index the batch...", like this one:

   ```
   2023-07-25 15:56:21,943 WARN  SolrIndexSubscriber - Failed to index the
    batch with ID '52'. Fix and re-ingest the data, and then use the admin/
   indexes/retryfailed REST endpoint to try again.
   ```

   The batch ID in the log file corresponds to the `batch_id` column in the database view.
3. Use the information in the `Indexing_Failed_Batches` database view to locate the incoming data that caused indexing to fail.

Resolving problems

To restore the search index to full health, you must fix the faulty data and then re-ingest and re-index it.

1. Correct the problem with the incoming data according to the information in the error messages.
2. Use your ETL pipeline to re-ingest the batch that caused the problem.
3. Call a second REST endpoint to retry indexing of the batch that failed on the previous attempt.

   The `POST /api/v1/admin/indexes/retryfailed` endpoint adds all failed batches to the queue for indexing.
4. After you call the endpoint, continue to monitor the indexing process as described above.

Note: Both REST endpoints require the logged-in user to have the `i2:Administrator:Indexing` permission in order to call them successfully.

Ingestion resources

The ingestion resources section contains information that is referenced elsewhere in the ingestion section.

Understanding the architecture

The physical architecture of your i2 Analyze deployment both affects and is affected by how you acquire and transform external data for the Information Store. Depending on the architecture, you might need to perform more deployment tasks before you can run data ingestion commands.

About this task

A complete solution for loading and ingesting data into the Information Store has four mandatory architectural components:

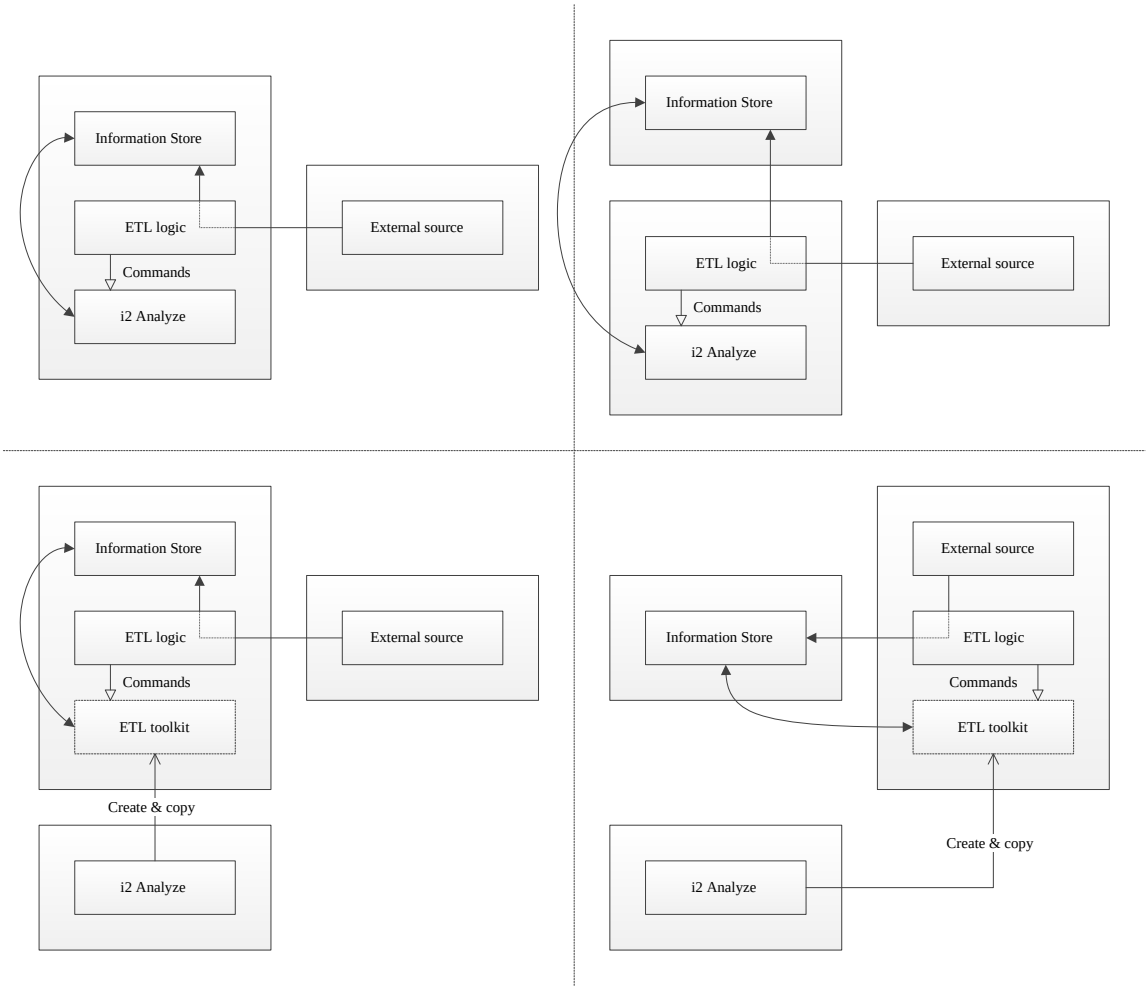
- The i2 Analyze server, and in particular the deployment toolkit that it contains
- The database management system that contains the Information Store and the staging tables
- An external data source
- The ETL logic that transforms the source data and loads it into the staging tables

Note: In addition to the mandatory components, you can opt to include a component that prepares your data for correlation, such as a matching engine or a context computing platform. These components can help you to determine when data in multiple different sources represents the same real-world "thing". This preparation of your data might occur as part of your ETL logic, or between an external data source and the ETL logic.

i2 Analyze supports physical architectures in which the database is hosted on the same server as the application, or on a different one. You can also choose to locate your ETL logic on the same server as the i2 Analyze application, or on the same server as the database, or on an entirely separate server.

The process of transforming source data can be demanding, especially if the requirements are complex or the volume is high. There are also scenarios in which you might want to automate the process of loading and then ingesting the external data. Ultimately, the architecture that you decide upon depends on the needs and constraints of your deployment.

The following diagram shows some of the permutations. The examples in the upper-left and upper-right quadrants represent deployments in which the ETL logic (implemented by a tool like IBM DataStage, for example) is co-hosted with the i2 Analyze application. The database can be on the same or a separate server; the solid arrows show data flow between the components during data load and ingestion.



The examples in the lower-left and lower-right quadrants represent deployments in which the ETL logic is on a separate server from the i2 Analyze application. (Typically, the ETL logic is hosted alongside the database or the external data source.) To enable the architecture, those deployments include the *ETL toolkit*, which is a cut-down version of the main deployment toolkit that targets only data ingestion.

When you need the ETL toolkit, you can generate it on the i2 Analyze server, and copy it to the server that hosts the ETL logic. When the ETL toolkit is properly configured, your ETL logic can run toolkit commands without reference to the rest of the deployment. If you decide to use the ETL toolkit, the next step is to deploy it. If not, you can move on to creating staging tables in the database.

Procedure

As the diagrams show, the ETL toolkit is most likely to be useful in deployments where the i2 Analyze application and the ETL logic are on separate servers. As you plan your approach to ingestion, consider the following:

- If the ETL logic is relatively simple and data volumes are low, there are benefits to colocating as many components as you can, especially in a new deployment.
- If your deployment requires separate servers from the start, or as it evolves over time, determine where the bottlenecks are. Is it limited by server speed or network speed?
- If the ETL logic is taxing a server that hosts other components, consider moving the logic, but be aware of the increase in network traffic.
- If the volume of data is taxing the network, consider colocating components when you are able. (You might not have permission to deploy components to some servers, for example.)

Results

By acting as a proxy for the i2 Analyze deployment toolkit, the ETL toolkit provides for more flexibility in your choice of architecture. In some circumstances you can separate the database, the ETL logic, and the i2 Analyze application without incurring a networking penalty.

Deploying the ETL toolkit

If your deployment includes logic that extracts, transforms, and loads data on a different server from the i2® Analyze application or the Information Store, consider deploying the ETL toolkit. The ETL logic can then run ETL toolkit commands to automate loading and ingesting data into the Information Store.

About this task

In an i2 Analyze deployment that uses data from an external source, the ETL logic is the processing that transforms source data for loading into the Information Store staging tables. In mature deployments, it is common for the ETL process to be automated so that loading and ingesting data happen in sequence, on a schedule.

When your ETL logic is colocated with the standard i2 Analyze deployment toolkit, the logic can use that toolkit to drive the ingestion process automatically. When those components are on separate servers, you can deploy the ETL toolkit to the server that hosts the ETL logic. The ETL toolkit provides the ingestion functions of the deployment toolkit in a stand-alone package.

Procedure

The ETL toolkit must be able to communicate with the Information Store with all the same credentials as the deployment toolkit. To enable this behavior, you use the deployment toolkit to create the ETL toolkit, and then copy it to the ETL logic server.

1. On the server that has the deployment toolkit, open a command prompt and navigate to the `toolkit\scripts` directory.
2. Run the `createEtlToolkit` command to generate the ETL toolkit:

   ```
   setup -t createEtlToolkit -p outputPath=output_path
   ```

   This command creates the ETL toolkit in a directory that is named `etltoolkit` in the `output_path` that you specify.
3. Copy the ETL toolkit to the server that hosts the ETL logic.

If the ETL logic and toolkit are on the same server as the database management system that hosts the Information Store, you do not need to modify the connection configuration. If the database management system is on a different server, then you must ensure that the ETL toolkit can communicate with the remote database.

1. Depending on your database management system, install the client tools for PostgreSQL, or Microsoft® Command Line Utilities for SQL Server, or Db2® client software on the server that hosts the ETL toolkit. For more information, see Installing a database.
2. Navigate to the `classes` directory of the ETL toolkit and open the `Connection.properties` file in a text editor.
3. Ensure that the value for the `db.installation.dir` setting is correct for the path to the PostgreSQL client tools or Microsoft Command Line Utilities for SQL Server or the Db2 client on the server that hosts this ETL toolkit.

   For example:

   ```
   db.installation.dir=C:/Program Files/IBM/SQLLIB
   ```
4. If you are using Db2 to host the Information Store, you must catalog the remote Db2 database. Run the following commands to enable the ETL toolkit to communicate with the Information Store:

   ```
   db2 catalog tcpip node node-name host-name server port-number
   db2 catalog database instance-name at node node-name
   ```

   Here, `host-name`, `port-number`, and `instance-name` are the values that are specified in the `topology.xml` file. `node-name` can be any value that you choose, but you must use the same value in both commands.

If the database management system that hosts the Information Store is not using SSL, then the process is complete.

If the database management system is configured to use SSL, you must also enable the ETL toolkit to communicate by using SSL. The detail of this process depends on your choice of database management system.

1. If you're using PostgreSQL, take the `i2-<database_management_system>-certificate.cer` certificate that you exported from the database management system when you configured SSL on the server that hosts the ETL toolkit, and store it in a convenient location on the server.
2. If you're using SQL Server or Db2:

   a. Register the `i2-<database_management_system>-certificate.der` certificate that you exported from the database management system when you configured SSL on the server that hosts the ETL toolkit.

      - On Windows, import the certificate into the Trusted Root Certification Authorities store for the current user.
      - On Linux, copy the certificate to the `/etc/pki/ca-trust/source/anchors` directory and use `update-ca-trust` to enable it as a system CA certificate.

   b. Create a truststore and import into the truststore the certificate that you exported from the database management system when you configured SSL.

      For example, run the following command:

      ```
      keytool -importcert -alias "dbKey"
              -file C:\i2\etltoolkit\i2-<database_management_system>-
      certificate.der
              -keystore "C:\i2\etltoolkit\i2-etl-truststore.jks"
              -storepass "<password>"
      ```

      Enter `yes` in response to the query, `Trust this certificate?`
3. Navigate to the `classes` directory of the ETL toolkit and open the `TrustStore.properties` file in a text editor.

4. If you're using PostgreSQL, populate the `DBTrustStoreLocation` property with the full path to the certificate that you stored earlier.
5. If you're using SQL Server or Db2:
   a. Populate the `DBTrustStoreLocation` and `DBTrustStorePassword` properties with the full path to the truststore that you created, and the password that is required to access it.

      For example:

      ```
      DBTrustStoreLocation=C:/i2/etltoolkit/i2-etl-truststore.jks
      DBTrustStorePassword=<password>
      ```

   b. You can use the Liberty profile `securityUtility` command to encode the password for the truststore.

      1. Navigate to the `bin` directory of the Open Liberty deployment that was configured by the deployment toolkit.
      2. In a command prompt, run `securityUtility encode` *`password`*, which generates and displays the encoded password. Use the entire value, including the `{xor}` prefix, for the `DBTrustStorePassword` property value. For more information about using the security utility, see securityUtility encode.

Results

The ETL toolkit is ready for use by your ETL logic to modify the Information Store. At key points in the processes of preparing for and performing ingestion, you can use commands in the ETL toolkit in place of deployment toolkit functions.

Information Store staging tables

At your request, i2 Analyze generates an Information Store staging table that can contain data for i2 Analyze records of a single entity type or link type. To generate the staging table, it uses information from the i2 Analyze schema, which is the same starting point from which it generates the main data tables during deployment.

An entity type staging table contains:

- At least one column for each property type in the schema.
- Columns for storing information that uniquely identifies data in its source. During ingestion, i2 Analyze can use this information to construct an origin identifier for the ingested data.
- Two columns to record when the data was created and updated in the source.
- Three columns for storing information that describes the source of the data. During ingestion, i2 Analyze can use this information to populate a source reference for the ingested data.
- Two columns to record the *correlation identifier type* and *correlation identifier key* of the data. During ingestion, i2 Analyze uses the information in these columns to construct the correlation identifier for the ingested data.
- A column for each security dimension that the security schema defines. During ingestion, i2 Analyze can use the information in these columns to implement per-record security.

For example, if the i2 Analyze schema contains this simplified entity type definition:

```
<EntityType Id="ET5" DisplayName="Person">
    <PropertyTypes>
        <PropertyType DisplayName="First (Given) Name"
                      LogicalType="SINGLE_LINE_STRING" Id="PER4"/>
        <PropertyType DisplayName="Birth place location"
```

```
                        LogicalType="GEOSPATIAL" Id="PER5"/>
        <PropertyType DisplayName="Date of Birth"
                        LogicalType="DATE" Id="PER9"/>
        <PropertyType DisplayName="Date and Time of Death"
                        LogicalType="DATE_AND_TIME" Id="PER10"/>
    </PropertyTypes>
</EntityType>
```

Then this SQL statement is the definition of a corresponding staging table in PostgreSQL:

```
CREATE TABLE is_staging.e_person (
        source_id character varying(50) COLLATE public.istore_collation,
        origin_id_type character varying(100) COLLATE
 public.istore_collation,
        origin_id_keys character varying(1000) COLLATE
 public.istore_collation,
        source_created timestamp without time zone,
        source_last_updated timestamp without time zone,
        correlation_id_type character varying(100) COLLATE
 public.istore_collation,
        correlation_id_key character varying(1000) COLLATE
 public.istore_collation,
        p_first_given_name character varying(250) COLLATE
 public.istore_collation,
        p_birth_place_location text COLLATE public.istore_collation,
        p_date_of_birth date,
        p0_date_and_time_of_deat timestamp(4) without time zone,
        p1_date_and_time_of_deat character varying(250) COLLATE
 public.istore_collation,
        p2_date_and_time_of_deat smallint,
        p3_date_and_time_of_deat timestamp(4) without time zone,
        security_level character varying(50) COLLATE
 public.istore_collation,
        security_compartment character varying(50) COLLATE
 public.istore_collation,
        source_ref_source_type character varying(200) COLLATE
 public.istore_collation,
        source_ref_source_location character varying(2000) COLLATE
 public.istore_collation,
        source_ref_source_image_url character varying(2000) COLLATE
 public.istore_collation
    );
```

This SQL statement is the definition of a corresponding staging table in SQL Server:

```
CREATE TABLE IS_Staging.E_Person(
        source_id nvarchar(50) NULL,
        origin_id_type nvarchar(100),
        origin_id_keys nvarchar(1000),
        source_created datetime2(6) NULL,
        source_last_updated datetime2(6) NULL,
        correlation_id_type nvarchar(100) NULL,
        correlation_id_key nvarchar(1000) NULL,
        p_first_given_name varchar(250) NULL,
        p_birth_place_location varchar(max) NULL,
        p0_date_and_time_of_deat datetime2(4) NULL,
        p1_date_and_time_of_deat nvarchar(250) NULL,
        p2_date_and_time_of_deat smallint NULL,
        p3_date_and_time_of_deat datetime2(6) NULL,
        security_level nvarchar(50) NULL,
        security_compartment nvarchar(50) NULL,
        source_ref_source_type nvarchar(200) NULL,
```

```
        source_ref_source_location nvarchar(2000) NULL,
        source_ref_source_image_url nvarchar(2000) NULL
    );
```

And this SQL statement is the definition of a corresponding staging table in Db2:

```
CREATE TABLE "IS_Staging"."E_Person" (
        "source_id" VARCHAR(50),
        "origin_id_type" VARCHAR(100),
        "origin_id_keys" VARCHAR(1000),
        "source_created" TIMESTAMP,
        "source_last_updated" TIMESTAMP,
        "correlation_id_type" VARCHAR(100),
        "correlation_id_key" VARCHAR(1000),
        "p_first_given_name" VARCHAR(250),
        "p_birth_place_location" VARCHAR(250),
        "p_date_of_birth" DATE,
        "p0_date_and_time_of_deat" TIMESTAMP,
        "p1_date_and_time_of_deat" VARCHAR(250),
        "p2_date_and_time_of_deat" SMALLINT,
        "p3_date_and_time_of_deat" TIMESTAMP,
        "security_level" VARCHAR(50),
        "security_compartment" VARCHAR(50),
        "source_ref_source_type" VARGRAPHIC(200),
        "source_ref_source_location" DBCLOB(2000),
        "source_ref_source_image_url" DBCLOB(2000)
    );
```

Note: Additionally, staging tables for link types contain a column for the direction of the link, and further columns for the information that uniquely identifies the link end data in the source.

The statements create the staging table in a separate schema from the Information Store data tables. Many of the columns in the staging table have names that are derived from the display names of the property types in the i2 Analyze schema. In most cases, the relationship between the schema and the staging table is obvious, but there are a number of extra columns and differences:

- The `source_id`, `origin_id_type`, `origin_id_keys` columns of the staging table can be used to store values that reference the rest of the data in its original source and can be used to make up the origin identifier of the resulting record.

  Note: If the staging table definition was for a link type, it would also contain `from_` and `to_` variations of each of the columns.
- The next two columns of the staging table are `source_created` and `source_last_updated`. You can use these columns to store information about when the data to be ingested was created and modified in its source.
- The next two columns of the staging table are `correlation_id_type` and `correlation_id_key`. If you want to correlate data during ingestion into the Information Store, you can use these columns to store values that i2 Analyze uses to generate correlation identifiers. For more information, see Overview of correlation.

  Note: Although populating the correlation identifier columns is not mandatory, doing so acts like a switch. The presence of correlation identifier values in any row of a staging table causes i2 Analyze to perform correlation for all the rows in that table.
- Any property type in the i2 Analyze schema that has the logical type `DATE_AND_TIME` occupies four columns in the staging table. These columns always appear in the same order:
  - The "`P0`" column is for the local date and time as originally recorded, as a `DATE_AND_TIME`.

- The "P1" column is for the time zone of the local date and time, as listed in the IANA database. For example, `Europe/London`.
- The "P2" column is for an indicator of whether Daylight Saving Time is (1) or is not (0) in effect.

  Note: i2 Analyze considers this value only when the time is ambiguous because it occurs during the hour that is "repeated" when Daylight Saving Time ends.
- The "P3" column is for the date and time as expressed in Coordinated Universal Time (UTC), as another `DATE_AND_TIME`.

For more information about the permitted values for `DATE_AND_TIME` columns in your database management system, see Information Store property value ranges.

- The next columns derive from the security schema rather than the i2 Analyze schema. One column exists for each security dimension that the security schema defines. You can use these columns if you want to give different dimension values to each i2 Analyze record that is created or updated as a result of ingestion.
- In link tables, there is also a `direction` column to store the direction of links.
- The final three columns are named `source_ref_source_type`, `source_ref_source_location`, and `source_ref_source_image_url`. These columns are used to populate the source reference that is generated when the data is ingested.

  For more information about implementing source references in your system, see Configuring source references.

The staging tables contain some, but never all, of the data for i2 Analyze records. They do not contain the type identifiers that Information Store records must have, and it is not mandatory to populate the columns for timestamps, security dimension values, or correlation identifiers. You can supply the remainder of the information in an ingestion mapping.

Information Store property value ranges

The Information Store places limits on the ranges of values that properties can contain. Different logical types in the i2 Analyze schema imply different limits, which are not always the same as the restrictions on the underlying database. It is important to consider the limits when you prepare data for ingestion.

Logical type limits that depend on the underlying database

Logical type	SQL Server values	PostgreSQL values	Db2 values
`SINGLE_LINE_STRING`	Up to 4000 bytes of UTF-16 characters. The default value is 250 *	Up to 8000 bytes of UTF-8 characters. The default value is 250 *	Up to 8000 bytes of UTF-8 characters. The default value is 250 *
`MULTI_LINE_STRING`	Up to 2^{31}-1 bytes of UTF-16 characters	Up to 2^{30} bytes of characters	Up to 32700 bytes of UTF-8 characters
`SELECTED_FROM`	Same as `SINGLE_LINE_STRING`	Same as `SINGLE_LINE_STRING`	Same as `SINGLE_LINE_STRING`
`SUGGESTED_FROM`	Same as `SINGLE_LINE_STRING`	Same as `SINGLE_LINE_STRING`	Same as `SINGLE_LINE_STRING`

* The value for `SINGLE_LINE_STRING` depends on the value that is specified for each property with that type in the i2 Analyze schema.

Logical type limits

Logical type	values
`DATE`	From **1753-01-01** to **9999-12-31**
`TIME`	From **00:00:00** to **23:59:59** *
`DATE_AND_TIME`	From **1753-01-01T23:00:00** ** to **9999-12-31T00:00:00**
`BOOLEAN`	**`true`** or **`false`**
`INTEGER`	From -2^{31} to $2^{31}-1$
`DOUBLE`	From 4.9×10^{-324} to $1.79769313486231 \times 10^{308}$ (Equivalent range for negative values. Maximum 15 digits of precision.)
`DECIMAL`	From **-999999999999999999.9999** to **999999999999999999.9999** (Maximum 18 digits before the decimal mark. Maximum 4 digits after it.)
`GEOSPATIAL`	From **-180.0000** to **180.0000** for longitude values, and **-90.0000** to **90.0000** for latitude values. The data must be in the Well-known text representation, `POINT(longitude latitude)`. The longitude and latitude values must conform to the WGS84 Bounds. For more information about the values, see Spatial reference - WGS84.

* If a Db2 database underlies the Information Store you can load time values that represent midnight as 24:00:00. When it stores such values, the database converts them to fit the ranges in the table.

** The *From* `DATE_AND_TIME` value starts at `23:00:00`.

Note:

- Data that includes non-printing or control characters might not be indexed and can cause errors.
- In addition to the values in the table, you can set the value of any non-mandatory property to null. In the staging table for an item type that has a `DATE_AND_TIME` property type, all four columns that the value is spread across must be null in that case.

Ingestion mapping files

An ingestion mapping file is an XML document whose structure is validated during the ingestion process. Every time that you instruct the Information Store to ingest data, you specify both the mapping

file to use, and the ingestion mapping within it. You can choose to put all your ingestion mappings in one file, or to spread them across several files.

Ingestion mappings have two complementary purposes. First, they make the association between an entity type or a link type in the i2 Analyze schema and a staging table in the database. Second, they provide any extra information that the Information Store requires but the staging tables do not contain.

For all record types, the extra information that an ingestion mapping can provide includes:

- The type identifier of the entity or link type that the mapping applies to
- The name of the data source that the data to be ingested comes from
- How to create an origin identifier for data of this type
- The security dimension values that all records of this type receive, if you do not use per-record security

Link type ingestion mappings provide further information that addresses the requirements of link records:

- The Information Store must be able to test that it already contains the entity records at the ends of an incoming link. The link type mapping must describe how to create the origin identifiers that are associated with those records so that the Information Store can look them up.
- To make the look-up more efficient, the link type mapping also contains the type identifiers of the entity records that appear at the "from" and "to" ends of the incoming links. A link type that can connect entities of several different types requires a separate mapping for each valid combination of end types.

Ingestion mapping syntax

The root element of an ingestion mapping file is an `<ingestionMappings>` element from the defined namespace. For example:

```
<ns2:ingestionMappings
    xmlns:ns2="http://www.i2group.com/Schemas/2016-08-12/IngestionMappings"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
...
</ns2:ingestionMappings>
```

Within the ingestion mapping file, you use an `<ingestionMapping>` element to define a mapping for a particular entity type or link type. Each `<ingestionMapping>` element has a mandatory `id` attribute that must be unique within the mapping file. You use the value to identify the mapping when you start ingestion. For example:

```
<ingestionMapping id="Person">
    ...
</ingestionMapping>
```

Note: For examples of complete ingestion mapping files, search for files with the name `mapping.xml` in the i2 Analyze deployment toolkit. All of those files contain definitions that are similar to the definitions here.

Entity type ingestion mappings

When the mapping is for an entity type, the `<ingestionMapping>` element has the following children:

`stagingArea`

The <stagingArea> element specifies where the mapping gets its staged data from. In this version of i2 Analyze, the staged data is always in a staging table, and <stagingArea> always has a <tableName> child.

- tableName

  The value of <tableName> is the name of the staging table that contains the data to be ingested.

For example:

```
...
<stagingArea xsi:type="ns2:databaseIngestionSource">
    <tableName>IS_Staging.E_Person</tableName>
</stagingArea>
...
```

itemTypeId

The value of the <itemTypeId> element is the identifier of the entity type (or the link type) to which the mapping applies, as defined in the i2 Analyze schema.

For example:

```
...
<itemTypeId>ET5</itemTypeId>
...
```

originId

The <originId> element contains a template for creating the origin identifier of each ingested row. <originId> has two mandatory child elements: <type> and <keys>.

For example:

```
...
<originId>
    <type>$(origin_id_type)</type>
    <keys>
        <key>$(origin_id_keys)</key>
    </keys>
</originId>
...
```

Here, $(origin_id_type) and $(origin_id_keys) are references to the columns named origin_id_type and origin_id_keys in the staging table to which this ingestion mapping applies. When the Information Store ingests the data, the values from the staging table become the origin identifier in the Information Store.

For more information about generating origin identifiers during ingestion, see Origin identifiers.

dataSourceName

The value of the <dataSourceName> element identifies the data source from which the data in the staging table came. It must match the name of an ingestion source that you provide to the Information Store during the ingestion process.

For example:

```
...
<dataSourceName>EXAMPLE</dataSourceName>
...
```

createdSource and lastUpdatedSource

By default, the ingestion process automatically puts the values from the `source_created` and `source_last_updated` columns of the staging tables into the Information Store. If you want to use the same values for all ingested data, you can override that behavior by including the non-mandatory `<createdSource>` and `<lastUpdatedSource>` elements and specifying values in the date-time string format for your database management system.

For example:

```
...
<createdSource>2002-10-04 09:21:33</createdSource>
<lastUpdatedSource>2002-10-05 09:34:45</lastUpdatedSource>
...
```

`securityDimensionValues`

Every row that the Information Store ingests must have at least one security dimension value from each dimension in the security schema. The Information Store staging tables contain a column for each access security dimension that the security schema defines.

In your ingestion process, you can use the staging table columns to store dimension values on a per-row basis. Alternatively, you can specify that all the data that the Information Store ingests through the same mapping get the same security dimension values.

In the ingestion mapping file, the `<securityDimensionValues>` element has `<securityDimensionValue>` children. For per-row security, use the value of each `<securityDimensionValue>` element to reference a security dimension column.

For example:

```
...
<securityDimensionValues>
    <securityDimensionValue>$(security_level)</securityDimensionValue>
    <securityDimensionValue>$(security_compartment)</securityDimensionValue>
</securityDimensionValues>
...
```

In the staging table, the referenced columns can contain either a single dimension value, or a comma-separated list of dimension values.

For per-mapping security, set the value of each `<securityDimensionValue>` element to a security dimension value.

For example:

```
...
<securityDimensionValues>
    <securityDimensionValue>HI</securityDimensionValue>
    <securityDimensionValue>UC</securityDimensionValue>
    <securityDimensionValue>OSI</securityDimensionValue>
</securityDimensionValues>
...
```

In either approach, the values that you specify must be present in the i2 Analyze security schema.

Link type ingestion mappings

When the ingestion mapping is for a link type, the `<ingestionMapping>` element has the same children that entity types require, plus the following ones:

`fromItemTypeId`

The value of the <fromItemTypeId> element is the type identifier of entity records that the schema permits at the "from" end of the link type to which this mapping applies.

For example:

```
...
<fromEntityTypeId>ET5</fromEntityTypeId>
...
```

fromOriginId

The <fromOriginId> element contains a template for creating the origin identifier of the entity record at the "from" end of each ingested link row. Its syntax is identical to the <originId> element.

The origin identifiers that result from <fromOriginId> must match the origin identifiers that result from the <originId> element for the entity type in question. The ingestion process uses this information to verify that the Information Store already ingested an entity record that has this origin identifier.

For example:

```
...
<fromOriginId>
    <type>$(from_origin_id_type)</type>
    <keys>
        <key>$(from_origin_id_keys)</key>
    </keys>
</fromOriginId>
...
```

For more information about generating origin identifiers during ingestion, see Origin identifiers.

toItemTypeId

The value of the <toItemTypeId> element is the type identifier of entity records that the schema permits at the "to" end of the link type to which this mapping applies.

For example:

```
...
<toEntityTypeId>ET10</toEntityTypeId>
...
```

toOriginId

The <toOriginId> element behaves identically to the <fromOriginId> element, except that it applies to the entity record at the "to" end of each ingested link row.

For example:

```
...
<toOriginId>
    <type>$(to_origin_id_type)</type>
    <keys>
        <key>$(to_origin_id_keys)</key>
    </keys>
</toOriginId>
...
```

For more information about generating origin identifiers during ingestion, see Origin identifiers.

linkDirection

The <linkDirection> element is a non-mandatory child of the <ingestionMapping> element. When you include a <linkDirection> element in an ingestion mapping, you can either provide the

same value for all links, or refer to the `direction` column of the staging table. Legal values for the element or the column are `WITH`, `AGAINST`, `BOTH`, and `NONE`.

For example, to use a fixed value:

```
...
<linkDirection>WITH</linkDirection>
...
```

Or, to use the value in the `direction` column:

```
...
<linkDirection>$(direction)</linkDirection>
...
```

If an ingestion mapping for a link type does not contain a `<linkDirection>` element, then any links that the Information Store ingests through the mapping have no direction.

References and system properties

In an ingestion mapping, you can use constants or references to specify values for i2 Analyze records. When you use a reference, the ingestion process retrieves a value from a staging table column or a property in a settings file. The settings file can also set system properties that control some aspects of ingestion into the Information Store.

By default, the settings file does not exist. You must create the settings file and specify the file when you run the ingestion command.

For example, you might call your settings file `ingestion_settings.properties` and it might contain the following name-value pairs:

```
SEC_LEVEL_VALUE=UC
SEC_COMPARTMENT_VALUE=HI,OSI
IngestionFailureMode=MAPPING
```

When you run the ingestion command, you reference your settings file as follows:

```
setup -t ingestInformationStoreRecords
      ...
      -p importConfigFile=ingestion_settings.properties
```

- System properties
- References

System properties

As well as providing values for ingestion mappings, you can use the settings file to configure the behavior of the ingestion process. The file supports a handful of system properties that you can set in the same way as you create and set custom properties.

`IngestionFailureMode [ RECORD | MAPPING ]`

When the Information Store encounters a problem with a record during ingestion, its default behavior is to log the error and move on to the next record. Failure is *record-based*. Instead, you can specify that a problem with one record causes the Information Store not to ingest any of the records from that staging table. Failure then is *mapping-based*.

In the settings file, the possible values for the `IngestionFailureMode` setting are `RECORD` or `MAPPING`. The default value is `RECORD`.

For example, to change the failure mode to mapping, add the following line to your settings file:

```
IngestionFailureMode=MAPPING
```

RecordFailureThreshold

During the ingestion process, if the number of errors that occur is greater than the value for the `RecordFailureThreshold` property, the process stops and no data is ingested into the Information Store. By default, the value for this property is 1000.

For example:

```
RecordFailureThreshold=500
```

IngestionRunstats [TRUE | FALSE]

During the ingestion process the RUNSTATS command updates statistics in the system catalog about the characteristics of a table, associated indexes, or statistical views.

In a table that contains many records, it can take the RUNSTATS command a long time to complete. If the number of records that you are ingesting is a small percentage of the total number of records in the table, it is recommended that you configure the ingestion process not to call the RUNSTATS command. You can then manually run the RUNSTATS command after a significant number of records are ingested.

You might also want to run your own RUNSTATS command instead of the one that is used by the ingestion process.

To prevent the ingestion process from calling the RUNSTATS command, set the value of `IngestionRunstats` to FALSE. For example:

```
IngestionRunstats=FALSE
```

The following commands are the RUNSTATS commands to run manually after an ingestion on each database when `IngestionRunstats` is set to false. Where `<table name>` is the table name where the data was ingested. For example, if you ingested Person data the table name is `IS_DATA.E_PERSON`.

For SQL Server:

```
ANALYZE <table name>
```

For PostgreSQL:

```
UPDATE STATISTICS <table name> WITH SAMPLE 10 PERCENT
```

For Db2:

```
RUNSTATS ON TABLE <table name> ON KEY COLUMNS AND INDEXES ALL ALLOW WRITE
 ACCESS TABLESAMPLE SYSTEM(10) INDEXSAMPLE SYSTEM(10)
```

The following command is a RUNSTATS command that you can run manually after an ingestion on large Db2 deployments. The command generates query output, and saves the output to the file named `ISStatisticsCollection.sql` that you can run.

```
db2 connect to ISTORE user <user name> using <password>

db2 -x
 "SELECT
      'RUNSTATS ON TABLE ' || TRIM(TABSCHEMA) ||'.'|| TRIM(TABNAME) || '
      WITH DISTRIBUTION ON ALL COLUMNS
      AND INDEXES ALL
      ALLOW WRITE ACCESS
      TABLESAMPLE BERNOULLI(10)
      INDEXSAMPLE BERNOULLI(10);'
 FROM SYSCAT.TABLES
 WHERE TYPE = 'T' AND TABSCHEMA = 'IS_DATA'"
```

```
> ISStatisticsCollection.SQL
```

`ImportBatchSize`

The `ImportBatchSize` controls the number of rows from a staging table that are ingested per batch. By default, the `ImportBatchSize` is set to 100,000.

For example:

```
ImportBatchSize=100000
```

If you are ingesting data into the system while analysts are using the system, it is possible that some analysis operations are blocked from returning results while a batch of data is being ingested. During the ingestion process, up to 100,000 rows (or the value of your batch size) can be locked in the database during each batch of the import. These locks cause a potential for the following analytical operations to stop returning results until the batch is complete: Find Path, Expand, and Visual Query.

To determine how long each batch takes to ingest, inspect the `IS_Data.Ingestion_Batches` table. The time that the batch takes to complete is the time that analytical operations might be blocked from returning results. If the time is too long for your requirements, you can reduce the value for `ImportBatchSize` to reduce the batch size and the time that it takes to complete.

For SQL Server, a batch size greater than 1,700 can cause table locks during link ingestion.

References

Many of the pieces of information that you provide in an ingestion mapping are fixed for that mapping. Item types, end types, and some parts of the origin identifier do not change between the i2 Analyze records that one mapping is responsible for. The most appropriate way to specify this kind of information is to use constant values on a per-mapping basis.

The two main reasons for preferring references to constant values lie at opposite ends of the spectrum:

- To give different values for the same field in records that are ingested through the same mapping, you can refer to a staging table column. This approach is appropriate for many non-property values that change from one record to the next.
- To use the same values across multiple ingestion mappings, refer to a property in a settings file. This approach might be appropriate when you want all the data from a source to get the same security dimension values. You can refer to the same property from every mapping that you write.

A settings file that defines properties for the ingestion process is just a text file that contains a set of *`name=value`* pairs, with one pair on each line:

```
SEC_LEVEL_VALUE=UC
SEC_COMPARTMENT_VALUE=HI,OSI
```

When you run one of the ingestion commands, you can supply it with the name of the properties file whose values you want to use.

To use a value by reference in an ingestion mapping, you use the `$(`*`name`*`)` syntax. *`name`* is the name of either a column in the staging table or a property in a settings file. For example, `$(SOURCE_ID)` and `$(DIRECTION)` refer to staging table columns, while in the previous example `$(SEC_LEVEL_VALUE)` and `$(SEC_COMPARTMENT_VALUE)` refer to properties.

Note: Since referring to columns and properties uses the same syntax, a clash can happen if a column and a property have the same name. In that case, the value of the property takes precedence.

Origin identifiers

The role of a source identifier is to reference the data for a record reproducibly in its original source. The source identifiers that records receive during ingestion are unique within i2 Analyze, and they have a special name in this context. They are called origin identifiers.

The nature of a origin identifier depends on the source and the creation method, and sometimes on whether the record is a link or an entity. When you ingest data into the Information Store, i2 Analyze compares the incoming origin identifier with existing records. If it finds a match, i2 Analyze updates a record instead of creating one.

After you develop your process for creating origin identifiers, you must continue to use that process. If you change the way that your origin identifiers are created and ingest the same data again, the Information Store creates new records for the data instead of updating the existing records. To ensure that changes to data are processed as updates, you must create your origin identifiers consistently.

For more information about different identifiers in i2 Analyze, see Identifiers in i2 Analyze records.

The structure of an origin identifier

During the ingestion process, you specify the data for your identifiers in the staging table and ingestion mapping file. An origin identifier is constructed of a "type" and "keys".

- **`type`**

  The "type" of an origin identifier allows the services in an i2 Analyze deployment to determine quickly whether they are interested in (or how to process) a particular row of data. The value of the type element does not have to be meaningful, but data from different sources generally have different values.

  For a deployment that uses a Db2 database, the length of the origin identifier type must not exceed 100 bytes, which is equivalent to 50 two-byte Unicode characters. For a SQL Server database, the limit is 200 bytes, or 100 two-byte Unicode characters.

- **`keys`**

  The "keys" of an origin identifier contain the information necessary to reference the data in its original source. The pieces of information that you use to make up the keys differs depending on the source of the data. For data that originates in relational sources, you might use keys whose values include the source name, the table name, and the unique identifier of the data within that table.

  The length of the origin identifier keys must not exceed the following sizes:

  - On PostgreSQL: 1000 Unicode characters.
  - On SQL Server: 692 bytes. This is equivalent to 346 Unicode characters.
  - On Db2: 1000 bytes. This is equivalent to 500 Unicode characters.

  It is recommended that your origin identifiers are as short as possible in length, and that any common values are at the end of the key.

  The content of your origin identifier keys must conform to Common Format and MIME Type for Comma-Separated Values (CSV) Files. i2 Analyze uses comma (' , ') as the delimiter and double-quote (' " ') as the quote character.

  - Do not include comma and double-quote characters next to each other in origin identifier keys. For example: `123",456`.

  Do not use non-printing or control characters in your origin identifiers because they might not be indexed correctly and cause your origin identifiers to be different from your intended values.

Creating origin identifiers for your data

There are two mechanisms for specifying the data for your origin identifiers. You can populate the staging table with all the information required to create the origin identifiers, or you can use a combination of information in the staging table and the ingestion mapping.

When you can provide all the information in the staging tables, there is less processing of the data during ingestion, which can improve ingestion performance.

All information in the staging table

If you can populate the staging table with all the information for your origin identifiers, you can use the `origin_id_type` and `origin_id_keys` columns to store this information. Populate the `origin_id_type` column with the type of your origin identifier. Populate the `origin_id_keys` column with a unique value that is already a composite of key values including the unique identifier from the source. When you use these columns, you must specify them in the ingestion mapping file that you use.

To ingest links, you must specify the origin identifiers at the end of the link. You specify the "to" end of the link in the `to_origin_id_type` and `to_origin_id_keys` columns, and the "from" end in `from_origin_id_type` and `from_origin_id_keys`.

When you specify all the information in the staging table, the origin identifier section of your ingestion mapping is more simple. For example:

```
...
<originId>
  <type>$(origin_id_type)</type>
  <keys>
    <key>$(origin_id_keys)</key>
  </keys>
</originId>
...
```

To specify the origin identifiers at the link ends:

```
...
<fromOriginId>
  <type>$(from_origin_id_type)</type>
  <keys>
    <key>$(from_origin_id_keys)</key>
  </keys>
</fromOriginId>

<toOriginId>
  <type>$(to_origin_id_type)</type>
  <keys>
    <key>$(to_origin_id_keys)</key>
  </keys>
</toOriginId>
...
```

Combination of information in the staging table and the ingestion mapping

If you cannot populate the staging table with all the information for your origin identifiers, you can use the `source_id` column in the staging table to contain the unique identifier from the source and provide any other keys and the origin identifier type in the ingestion mapping.

To ingest links, you must specify the origin identifiers at the end of the link. You specify the unique identifier of the "to" end of the link in the `to_source_id` column, and the "from" end in

`from_source_id`. In the ingestion mapping, you specify the other keys that make up the origin identifiers of the link ends.

When you provide the information in both the staging table and the ingestion mapping, the mapping file is more complex. For example:

```
...
<originId>
  <type>OI.EXAMPLE</type>
  <keys>
    <key>$(source_id)</key>
    <key>PERSON</key>
  </keys>
</originId>
...
```

To specify the origin identifiers at the link ends, if the to end is an "Account" entity type:

```
...
<fromOriginId>
  <type>OI.EXAMPLE</type>
  <keys>
    <key>$(from_source_id)</key>
    <key>PERSON</key>
  </keys>
</fromOriginId>

<toOriginId>
  <type>OI.EXAMPLE</type>
  <keys>
    <key>$(to_source_id)</key>
    <key>ACCOUNT</key>
  </keys>
</toOriginId>
...
```

The `ingestInformationStoreRecords` task

The parameters and usage of the `ingestInformationStoreRecords` depends on whether you are using the i2 Analyze deployment toolkit or the ETL toolkit.

i2 Analyze deployment toolkit

The deployment toolkit command for ingesting records looks like this:

```
setup -t ingestInformationStoreRecords
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p importMode=STANDARD|VALIDATE|BULK
```

Here, `ingestion_mapping_file` is the path to the XML file that contains the mapping that you want to use, and `ingestion_mapping_id` is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The `importLabel`, `importConfigFile`, and `importMode` parameters are optional:

- When you specify `importLabel`, `ingestion_label` is a name that identifies a particular use of the command in the Information Store's `IS_Public.Ingestion_Deletion_Reports` view.

- When you specify `importConfigFile`, `ingestion_settings_file` is the path to a settings file that contains `name=value` pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- When you specify `importMode`, you can set it to `STANDARD`, `VALIDATE`, or `BULK`.
  - `STANDARD` mode can be used to ingest new and updated records, with or without correlation. If you do not specify `importMode`, `STANDARD` mode is used.
  - `VALIDATE` mode checks the validity of the specified mapping, but no ingestion takes place.
  - `BULK` mode can be used to ingest new records without correlation.

ETL toolkit

The equivalent ETL toolkit command looks like this. The ETL toolkit command can also delete provenance from the Information Store.

```
ingestInformationStoreRecords
      -imf  ingestion_mapping_file
      -imid ingestion_mapping_id
      -il   ingestion_label
      -icf  ingestion_settings_file
      -lcl  true|false
      -im   STANDARD|VALIDATE|BULK|DELETE_PREVIEW|DELETE|BULK_DELETE
```

Here, `ingestion_mapping_file` is the path to the XML file that contains the mapping that you want to use, and `ingestion_mapping_id` is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The other parameters are optional:

- When you specify `il`, `ingestion_label` is a name that identifies a particular use of the command in the Information Store's `IS_Public.Ingestion_Deletion_Reports` view.
- When you specify `icf`, `ingestion_settings_file` is the path to a settings file that contains `name=value` pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- When you specify `lcl` as `true`, any links that are removed as part of an entity delete operation are logged in `IS_Public.D<import identifier><entity type id>_<link type id>_Links` tables. For example, `IS_Public.D20180803090624143563ET5_LAC1_Links`.

  You can specify `lcl` with the `DELETE` import mode only.
- When you specify `im` you can set it to one of the following modes:
  - `STANDARD` mode can be used to ingest new and updated records, with or without correlation. If you do not specify `im`, `STANDARD` mode is used.
  - `VALIDATE` mode checks the validity of the specified mapping, but no ingestion takes place.
  - `BULK` mode can be used to ingest new records without correlation.
  - `DELETE_PREVIEW` mode can be used to preview the effect of running a delete.
  - `DELETE` mode can be used to delete provenance from the Information Store.
  - `BULK_DELETE` mode can be used to delete provenance that does not contribute to a correlated record.

The previewDeleteProvenance and deleteProvenance tasks

To update the Information Store for deleted data in an external source, you update the Information Store by using staging tables and the deployment toolkit.

In the i2 Analyze deployment toolkit, you can use the `previewDeleteProvenance` and `deleteProvenance` toolkit tasks to update the Information Store. If you are using the ETL toolkit, you use the `ingestInformationStoreRecords` command with different import modes. For more information about using the ETL toolkit to remove provenance, see the *ETL toolkit* section in The `ingestInformationStoreRecords` task.

Because the delete process might cause significant numbers of i2 Analyze records to be deleted, two commands are provided. The first command previews the effect of running the second command before you commit to doing so. In the deployment toolkit, the two commands have different names but the same syntax:

```
setup -t previewDeleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
```

```
setup -t deleteProvenance
      -p importMappingsFile=ingestion_mapping_file
      -p importMappingId=ingestion_mapping_id
      -p importLabel=ingestion_label
      -p importConfigFile=ingestion_settings_file
      -p logConnectedLinks
      -p importMode=BULK_DELETE
```

Here, `ingestion_mapping_file` is the path to the XML file that contains the mapping that you want to use, and `ingestion_mapping_id` is the identifier of the mapping within that file. The latter is mandatory unless the file contains only one mapping.

The other parameters are optional:

- When you specify `importLabel`, `ingestion_label` is a name that identifies a particular use of the command in the Information Store's `IS_Public.Ingestion_Deletion_Reports` view.
- When you specify `importConfigFile`, `ingestion_settings_file` is the path to a settings file that contains `name=value` pairs. You can refer to names in the settings file from references in the ingestion mapping file to use their values when you run the `ingestInformationStoreRecords` command.
- When you specify `logConnectedLinks`, any links that are removed as part of an entity delete operation are logged in `IS_Public.D<import identifier><entity type id>_<link type id>_Links` tables. For example, `IS_Public.D20180803090624143563ET5_LAC1_Links`.

  You can specify `logConnectedLinks` when you do not specify `importMode` only.
- When you specify `importMode`, you can set it to `BULK_DELETE`. `BULK_DELETE` mode can be used to delete provenance that does not contribute to a correlated record. If you do not specify `importMode`, the standard delete process is used.

Understanding ingestion reports

Every attempt to add, update, or delete data in the Information Store through the deployment or ETL toolkit adds rows to the `IS_Public.Ingestion_Deletion_Reports` view. You can use the

contents of this view to track the history of all such operations, and to examine the impact of a particular operation.

About this task

Each time that you run a command that might change the contents of the Information Store, you create a job in the database. Each job acts on one or more batches of i2 Analyze records. There is always one batch per item type that the command affects, but there can also be several batches for the same type if the number of affected records is large.

For example, consider a command that processes updates for deleted Person entity data. The first batch in the resulting job is for Person records, and there might be more such batches if there are many records to be deleted. If the Person data has links, then the job has further batches for each type of link that might get deleted as a result of the entity deletion.

The `IS_Public.Ingestion_Deletion_Reports` view contains information about every batch from every toolkit operation to create or update data in the Information Store. When you query the view, include **`ORDER BY job_id`** to group entries for the same job.

Note: Deletion-by-rule operations also result in job and batch creation, and view population, according to the same rules. For more information, see the Deletion Guide.

The first few columns in the view have the same value for all batches within a job:

Column name	Description
`label`	The value that you passed in the `importLabel` parameter of a toolkit command, or the value that a deletion-by-rule operation generates, or null.
`job_id`	The server-assigned identifier for this ingestion or deletion job. This identifier is also a cross-reference to the `Deletion_By_Rule_Log` view if the job originated from a deletion-by-rule operation.
`ingestion_mode`	The value that you passed in the `importMode` parameter, or **Delete** for all deletion-by-rule operations.
`validation_mode`	A description of how the job was configured to react to errors during the operation.
`error_threshold`	The threshold that applies to some of the validation modes.
`primary_item_type`	The i2 Analyze schema ID of the item type that was specified at job creation.
`primary_record_count`	The number of records of the primary item type that were affected by the job. (Deleting entity data can affect link records too.)

Column name	Description
`start_time`	The start time of the job as a whole.
`end_time`	The end time of the job as a whole.

The remaining columns can have different values for different batches of records:

Column name	Description
`batch_item_type`	The i2 Analyze schema ID of the item type that was acted on in this batch. For at least one batch, the `batch_item_type` is the same as the `primary_item_type`.
`batch_start_time`	The start time of this batch, which is always later than the start time of the job.
`batch_end_time`	The end time of this batch, which is always earlier than the end time of the job.
`insert_count`	The number of rows of data from this batch that were inserted to the Information Store, resulting in new i2 Analyze records.
`update_count`	The number of rows of data from this batch that updated existing records in the Information Store.
`merge_count`	The number of *merge* operations that occurred in the Information Store from this batch.
`unmerge_count`	The number of *unmerge* operations that occurred in the Information Store from this batch.
`delete_count`	The number of pieces of provenance that were deleted from the Information Store as a result of this batch.
`delete_record_count`	The number of records that were deleted from the Information Store as a result of this batch.
`reject_count`	The number of rows that were rejected from this batch during processing because they are invalid.
`status`	An indicator of the result of this batch, from success (all rows processed correctly) through partial success to failure (no rows processed).

Column name	Description
`reject_view`	The full name of the view that contains details of any rejected rows.
`stack_trace`	If i2 Analyze generated a stack trace as a result of errors during ingestion or deletion, this column contains it.

Examples

Ingest example

The (abbreviated) report for successful ingestion operations might look like this:

job_id	**1**	**2**
`ingestion_mode`	Standard	Standard
`primary_item_type`	ET10	ET4
`primary_record_count`	62	8
`batch_item_type`	ET10	ET4
`batch_start_time`	2017-11-30 15:27:06.76	2017-11-30 15:27:09.45
`batch_end_time`	2017-11-30 15:27:09.87	2017-11-30 15:27:09.63
`insert_count`	57	7
`update_count`	0	0
`merge_count`	5	1
`unmerge_count`	0	6
`delete_count`	0	0
`delete_record_count`	0	0
`reject_count`	0	0
`status`	Succeeded	Succeeded

In this example, several commands to ingest entity records resulted in the creation of several jobs. Each job demonstrates different behavior that is possible during ingestion, including correlation operations:

- **`JOB_ID` `1`**

This job demonstrates what the ingestion report can look like when data in the staging table causes merge operations. In this example, five merge operations are completed on the incoming rows of data, as shown in the `merge_count` column. This results in 57 i2 Analyze records created from the 62 rows of data, as shown in the `insert_count` and `primary_record_count` columns. This includes merging five rows of data with existing i2 Analyze records in the Information Store.

- **`JOB_ID 2`**

  This job demonstrates what the ingestion report can look like when the data in the staging table causes unmerge and merge operations. In this example, six unmerge operations are completed on the incoming rows of data, as shown in the `unmerge_count` column. One merge operation is completed on the incoming rows, as shown in the `merge_count` column. This results in 7 i2 Analyze records created, from eight rows of data as shown in the `insert_count` and `primary_record_count` columns. The `primary_record_count` value does not include the `unmerge_count`.

Delete example

The (abbreviated) report for a successful delete operation might look like this:

	Batch 1	**Batch 2**	**Batch 3**	**Batch 4**	**Batch 5**
`job_id`	26	26	26	26	26
`ingestion_mode`	Delete	Delete	Delete	Delete	Delete
`primary_item_type`	ET5	ET5	ET5	ET5	ET5
`primary_record_count`	324	324	324	324	324
`batch_item_type`	ET5	LAC1	LAS1	LEM1	LIN1
`batch_start_time`	2017-11-30 15:27:06.76	2017-11-30 15:27:08.60	2017-11-30 15:27:08.60	2017-11-30 15:27:09.43	2017-11-30 15:27:09.45
`batch_end_time`	2017-11-30 15:27:09.87	2017-11-30 15:27:09.30	2017-11-30 15:27:09.29	2017-11-30 15:27:09.62	2017-11-30 15:27:09.63
`insert_count`	0	0	0	0	0
`update_count`	0	0	0	0	0
`merge_count`	0	0	0	0	0
`unmerge_count`	0	0	0	0	0
`delete_count`	324	187	27	54	33
`delete_record_count`	320	187	27	54	33
`reject_count`	0	0	0	0	0

	Batch 1	Batch 2	Batch 3	Batch 4	Batch 5
status	Succeeded	Succeeded	Succeeded	Succeeded	Succeeded

In this example, a command to update the Information Store for deleted entity data (with item type ET5) resulted in the creation of a job with five batches. The first few columns of the `Ingestion_Deletion_Reports` view contain the same values for all batches in the same job. Later columns reveal how deleting entity records results in the deletion of connected link records (with item types LAC1, LAS1, LEM1, LIN1).

In one case, the `delete_record_count` value is less than the `delete_count` value. This is because some of the provenance to be deleted was associated with an i2 Analyze record that had more than one piece of provenance. An i2 Analyze record is deleted only when the last associated provenance is deleted.

Troubleshooting the ingestion process

The commands that you run during the ingestion process send information about their progress to the command line and a log file. If any command encounters errors or does not run to completion, you can read the output to help you to diagnose the problem.

When an ingestion process runs to completion, the final output from the command is a report of what happened to the Information Store. The reports appear on the command line and in the ingestion log at `toolkit\configuration\logs\importer\i2_Importer.log`. The three possible end states are success, partial success, and failure.

- **Success**

  If the ingestion command processed all of the rows in the staging table without error, then the Information Store reflects the contents of the staging table. The command reports success like this example:

```
> INFO  [IImportLogger] - Total number of rows processed: 54
> INFO  [IImportLogger] - Number of records inserted: 0
> INFO  [IImportLogger] - Number of records updated: 54
> INFO  [IImportLogger] - Number of merges: 0
> INFO  [IImportLogger] - Number of unmerges: 0
> INFO  [IImportLogger] - Number of rows rejected: 0
> INFO  [IImportLogger] - Duration: 5 s
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Result: SUCCESS
```

- **Partial success**

  If you ran the command in record-based failure mode, and it processed some of the rows in the staging table without error, then it reports partial success like this example:

```
> INFO  [IImportLogger] - Total number of rows processed: 34
> INFO  [IImportLogger] - Number of records inserted: 0
> INFO  [IImportLogger] - Number of records updated: 30
> INFO  [IImportLogger] - Number of merges: 0
> INFO  [IImportLogger] - Number of unmerges: 0
> INFO  [IImportLogger] - Number of rows rejected: 4
> INFO  [IImportLogger] - Duration: 4 s
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Result: PARTIAL SUCCESS
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Total number of errors: 4
> INFO  [IImportLogger] - Error categories:
```

```
> INFO  [IImportLogger] - ABSENT_VALUE: 4
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - The rejected records and errors are recorded in
 the database. For details, use the following view:
> INFO  [IImportLogger] - IS_Staging.S20171204122426717092ET5_Rejects_V
```

The records in the Information Store reflect the rows from the staging table that the command successfully processed. The report includes the name of a database view that you can examine to discover what went wrong with each failed row.

- **Failure**

  If you ran the command in mapping-based failure mode, then any error you see is the first one that it encountered, and the report is of failure:

```
> INFO  [IImportLogger] - Total number of rows processed: 1
> INFO  [IImportLogger] - Number of records inserted: 0
> INFO  [IImportLogger] - Number of records updated: 0
> INFO  [IImportLogger] - Number of merges: 0
> INFO  [IImportLogger] - Number of unmerges: 0
> INFO  [IImportLogger] - Number of rows rejected: 0
> INFO  [IImportLogger] - Duration: 0 s
> INFO  [IImportLogger] -
> INFO  [IImportLogger] - Result: FAILURE
```

  When the process fails in this fashion, the next lines of output describe the error in more detail. In this event, the command does not change the contents of the Information Store.

Note: If a serious error occurs, it is possible for the ingestion command not to run to completion. When that happens, it is harder to be certain of the state of the Information Store. The ingestion process uses batching, and the records in the store reflect the most recently completed batch. If you are using the bulk import mode, see Bulk import mode error for more information about recovering from errors at this stage.

If the command reports partial success, you might be able to clean up the staging table by removing the rows that were ingested and fixing the rows that failed. However, the main benefit of record-based failure is that you can find out about multiple problems at the same time.

The most consistent approach to addressing failures of all types is to fix up the problems in the staging table and run the ingestion command again. The following sections describe how to react to some of the more common failures.

Link rows in the staging table refer to missing entity records

When the Information Store ingests link data, you might see the following error message in the console output:

```
Link data in the staging table refers to missing entity records
```

This message is displayed if the entity record at either end of a link is not present in the Information Store. To resolve the error:

- Examine the console output for your earlier operations to check that the Information Store ingested all the entity records properly.
- Ensure that the link end origin identifiers are constructed correctly, and exist for each row in the staging table.
- Ensure that the link type and the entity types at the end of the links are valid according to the i2 Analyze schema.

Then, rerun the ingestion command.

Rows in the staging table have duplicate origin identifiers

During any ingestion procedure, but especially when a staging table is large, you might see the following error message in the console output:

```
Rows in the staging table have duplicate origin identifiers
```

This message is displayed when several rows in a staging table generate the same origin identifier. For example, more than one row might have the same value in the `source_id` column.

If more than one row in the staging table contains the same provenance information, you must resolve the issue and repopulate the staging table. Alternatively, you can separate the rows so that they are not in the same staging table at the same time.

This problem is most likely to occur during an update to the Information Store that attempts to change the same record (with the same provenance) twice in the same batch. It might be appropriate to combine the changes, or to process only the last change. After you resolve the problem, repopulate the staging table and rerun the ingestion command.

Geospatial data is in the incorrect format

During an ingestion procedure that contains geospatial data, you might see the following error messages in the console output:

On PostgreSQL:

```
ERROR: parse error - invalid geometry Hint: "FO" <-- parse error at position
 2 within geometry.
```

On SQL Server:

```
System.FormatException: 24114: The label FOO(33.3 44.0) in the input well-
known text (WKT) is not valid.
```

On Db2:

```
SQLERRMC=GSEGEOMFROMWKT;;GSE3052N  Unknown type "FOO(33.3" in WKT.
```

This message is displayed when data in a geospatial property column is not in the correct format.

Data in geospatial property columns must be in the `POINT(longitude latitude)` format. For more information, see Information Store property value ranges.

Error occurred during a correlation operation

During an ingestion procedure with correlated data, you might see the following error message in the console output:

```
An error occurred during a correlation operation. There might be some data
 in an unusable state.
```

This message is displayed if the connection to the database or Solr is interrupted during a correlation operation.

To resolve the problem, you must repair the connection that caused the error, and then run the `syncInformationStoreCorrelation` toolkit task. This task synchronizes the data in the Information Store with the data in the Solr index so that the data returns to a usable state.

After you run the `syncInformationStoreCorrelation` task, reingest the data that you were ingesting when the failure occurred. Any attempt to run an ingestion or a deletion command before you run `syncInformationStoreCorrelation` will fail.

Ingestion with correlated data is still in progress

During an ingestion procedure, you might see the following error message in the console output:

```
You cannot ingest data because an ingestion with correlated data is still in
 progress,
or because an error occurred during a correlation operation in a previous
 ingestion.
```

If another ingestion is still in progress, you must wait until it finishes. If a previous ingestion failed during a correlation operation, you must run the `syncInformationStoreCorrelation` toolkit task.

For more information about running the `syncInformationStoreCorrelation` toolkit task, see Error occurred during a correlation operation.

Ingestion of the same item type is still in progress

During an ingestion procedure, you might see the following error message in the console output:

```
You cannot ingest data for item type <ET5> because an ingestion is still in
 progress.
You must wait until the process is finished before you can start another
 ingestion for this item type.
```

If another ingestion of the same item type is still in progress, you must wait until it finishes.

If you are sure that the ingestion is complete or not in progress, you can remove the file that is blocking the ingestion. To determine whether an ingestion is in progress, a file is created in the temporary directory on the server where the ingestion command was run. For example, `AppData\Local\Temp`. The file name is `INGESTION_IN_PROGRESS_<item type ID>`. After you remove the file, you can run the ingestion command again.

Bulk import mode error

The symptoms of this type of failure are a stack trace and failure message in the console and importer log. To recover from a failure at this time:

1. Identify the cause of the failure. You must use the SQL error codes to determine the cause of the failure.

   You might see error messages about the following issues:

   - Log size or connectivity issues.
   - Invalid data in the staging table.
2. Fix the problem that caused the failure. This might include ensuring connectivity to the database or increasing the log size.
3. After you resolve the problem that caused the error, you can attempt the ingestion again. If any of the rows in the staging table were already ingested into the Information Store, you must remove them from the staging table before you can ingest in bulk mode.

   - In the console or importer log, if the value for `Number of rows accepted` is 0 then run the ingestion command again.
   - In the console or importer log, if the value for `Number of rows accepted` is greater than 0, you must ensure that these records are not ingested again.

     Before you run the ingestion command again, add the `CheckExistingOriginIds=filter` setting to the import configuration file. When this value is set, the ingestion process calculates

whether the origin identifiers in the staging table already exist in the Information Store and does not attempt to ingest them again.

When this is set to `filter`, the ingestion might take longer to complete. After the ingestion that failed is complete, you can remove the `CheckExistingOriginIds` setting from your import configuration file for future ingestion operations.

For more information about creating an import configuration file, see References and system properties.

Staging table already exists

When you run the `createInformationStoreStagingTable` toolkit task, the following stack trace is displayed:

```
Exception in thread "main" java.lang.IllegalStateException: Table
 IS_STAGING.XXX already exists
...
:createInformationStoreStagingTable FAILED

FAILURE: Build failed with an exception.
```

Where XXX is the name of the table specified for the `tableName` argument. For example, `E_Person`.

This error is caused because a staging table with the same name already exists. You can use this existing table for your ingestion or you can remove it and recreate it.

If you have modified the item type in the schema, you must recreate the staging table to update it with the schema changes. To recreate the staging table, you must delete it from the database before using the toolkit command to recreate.

To delete the staging table, connect to your database management system and issue a `drop table` command.

For example `DROP TABLE IS_STAGING.E_PERSON`.

For more information, see:

- SQL Server - DROP TABLE
- PostgreSQL - DROP TABLE
- Db2 - Dropping tables

Information Store data correlation

This documentation provides an overview of the correlation functions that are available during data ingestion into the Information Store in i2 Analyze. Later sections describe how to configure correlation with an example use case.

Intended audience

This documentation is intended for users who want to correlate data as it is being ingested into the Information Store.

Users must understand how to ingest data into the Information Store. For more information, see Information Store data ingestion.

Users must understand the i2 Analyze data model, and i2 Analyze record structure. For more information, see Data in i2 Analyze records.

Overview of correlation

Correlation is the process of associating multiple pieces of data with each other based on strong identifiers. For the process of ingesting data into the Information Store, i2 Analyze can use correlation identifiers that you provide to determine how to process and represent data in i2 Analyze records.

Correlation in i2 Analyze

During the ingestion process for the Information Store, correlation can be used to determine when data that is ingested is associated with existing records, and must be represented by a single i2 Analyze record. Conversely, if data no longer represents the same object, the data can be represented by multiple i2 Analyze records. In i2 Analyze, these operations are known as *merge* and *unmerge*.

During the correlation process, an identifier is used to determine how each row of data is associated. You present the identifier to the Information Store with the other staging data during ingestion.

You can limit the use of correlation to data from a specific source, of certain item types, or per row of data ingested into the Information Store. You do not have to provide a correlation identifier for all the data that you ingest into the Information Store.

Correlation uses

You might want to use correlation when you are ingesting data that originates from disparate sources, which have common properties, or have the potential to represent the same real-world objects. For example, if you have two data sources that contain information about people.

Another scenario where you might use correlation, is when the data that you are ingesting is in the form of event driven models (crime or complaint reports) where the same actors (people, locations, phones, and vehicles) might be referred to frequently in the same source.

Correlation can be used in these scenarios to combine multiple source records into single i2 Analyze records for link analysis.

Correlation method

In i2 Analyze, correlation identifiers and implicit discriminators are used to determine how the Information Store processes data during ingestion.

When you ingest data into the Information Store, you can provide a correlation identifier type and key value that are used to construct the correlation identifier for each row of data in the staging table. The type and key values that you provide are used to process data that is determined to represent the same real world object. Implicit discriminators are formed from parts of the i2 Analyze data model in the Information Store. Even if correlation identifiers match, if values for elements of the i2 Analyze data model are not compatible, that data cannot be represented by the same i2 Analyze record. For more information about correlation identifiers and implicit discriminators, see Correlation identifiers.

During the ingestion process, i2 Analyze compares the correlation identifiers of the data to be ingested and existing data in the Information Store. The value of the correlation identifiers determines the operations that occur. For more information about the correlation operations that can occur, see Correlation operations.

The example data sets demonstrate the correlation behavior in this release of i2 Analyze. For more information, see Ingesting example correlation data.

Correlation identifiers

The role of a correlation identifier is to indicate that data is about a specific real-world object. If multiple pieces of data are about the same specific real-world object, they have the same correlation identifier. At ingestion time, the correlation identifier of incoming data informs the Information Store how to process that data. Depending on the current state of the i2 Analyze record that is associated with the incoming data, a match with the correlation identifier on an inbound row of data determines the outcome of the association.

You specify the values for the correlation identifier in the staging table that you are ingesting the data from. The correlation identifier is made up of two parts, the *correlation identifier type* and the *correlation identifier key*.

- **type**

  The *type* of a correlation identifier specifies the type of correlation key that you are using as part of the correlation identifier. If you are generating correlation keys by using different methods, you might want to distinguish them by specifying the name of the method as the correlation identifier type. If your correlation keys are consistent regardless of how they are created, you might want to use a constant value for the correlation identifier type.

  When you specify the identifier type, consider that this value might be seen by analysts.

  The length of the value for the *type* must not exceed 100 bytes. This value is equivalent to 50 Unicode characters.

- **key**

  The *key* of a correlation identifier contains the information necessary to identify whether multiple pieces of data represent the same real-world object. If multiple pieces of data represent the same real-world object, they have the same correlation identifier key.

  The length of the correlation identifier key must not exceed the following sizes:

  - On PostgreSQL: 1000 Unicode characters.
  - On SQL Server: 692 bytes. This is equivalent to 346 Unicode characters.
  - On Db2: 1000 bytes. This is equivalent to 500 Unicode characters.

To prepare your data for correlation by i2 Analyze, you might choose to use a matching engine or context computing platform. Matching engines and context computing platforms can support the identification of matches that enable you to identify when data that is stored in multiple sources represents a single entity. You can provide these values to the Information Store at ingestion time.

For example, IBM InfoSphere Identity Insight provides resolved entities with an entity identifier. If you are using such a platform, you can populate the correlation identifier type to record this, for example `identityInsight`. You might populate the correlation identifier key with the entity identifier, for example `1234`. This generates a correlation identifier of `identityInsight.1234`.

Alternatively, as part of the data processing to add data to the staging tables, you might populate the correlation identifier with values from property fields that distinguish entities. For example, to distinguish People entities you might combine the values for their date of birth and an identification number, and you might specify the type as `manual`. This generates a correlation identifier of `manual.1991-02-11123456`.

The complete correlation identifier is used for comparison. Only data with correlation identifiers of the same type is correlated.

For more information about specifying a correlation identifier during the ingestion process, see Information Store staging tables.

Implicit discriminators

In addition to the correlation identifier that is created from the type and key values that you provide, *implicit discriminators* are also used during the matching process. In addition to the correlation identifier, the following implicit discriminators are also compared. The implicit discriminators must be compatible to enable correlation to occur.

Item type

The item type of the data that you are ingesting must be the same as the item type of the existing i2 Analyze record that is matched by the correlation identifier. If the item types are not the same, then no correlation operations occur.

Security dimension values

The security dimension values of the data that you are ingesting must be the same as the data that is matched by the correlation identifier. If the security dimension values are not the same, then no correlation operations occur.

Link direction and ends

For link data, the link direction and ends of the data that you are ingesting must be the same as the data that is matched by the correlation identifier. If the link direction and ends are not the same, then no correlation operations occur.

The direction and ends of a link are inspected, and direction is respected. For example, a link from A to B of direction 'WITH' matches with a link from B to A of direction 'AGAINST'. A link from A to B of direction 'WITH' does not match with a link from A to B of direction 'AGAINST'.

Correlation operations

When the Information Store receives a correlation identifier, the way the system responds depends on the value of the correlation identifier, and the state of any records that are associated with it.

The following section explains the *merge* and *unmerge* operations that the system can respond with when it receives a correlation identifier. This response is in addition to the insert and update operations that are part of the standard ingestion process.

Merge

When one or more pieces of data are determined to represent the same real-world object, the data is merged into a single i2 Analyze record. For the Information Store to merge data, the correlation identifiers must match, the implicit discriminators must be compatible, and the origin identifiers must be different.

During ingestion, a merge operation can occur in the following scenarios:

- New data in the staging table contains the same correlation identifier as an existing record in the Information Store. The new data has an origin identifier that is not associated with the existing record.
- An update to an existing record with a single piece of provenance in the Information Store causes the correlation identifier of that record to change. The new correlation identifier matches with another record in the Information Store.
- Multiple rows of data in the staging table contain the same correlation identifier. The Information Store ingests the data as a new i2 Analyze record, or the record merges with an existing record in the Information Store.

After a merge operation, the following statements are true for the merged record:

- The record has a piece of provenance for all of the source information that contributed to the merged record.
- By default, the property values for the merged i2 Analyze record are taken from the source information associated with the provenance that has the most recent value for the `source_last_updated` column.

  If only one piece of provenance for a record has a value for the `source_last_updated` column, the property values from the source information that is associated with that provenance are used. Otherwise, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The piece of provenance that is last in the order is chosen. To ensure data consistency, update your existing records with a value for the `source_last_updated` column before you start to use correlation, and continue to update the value.

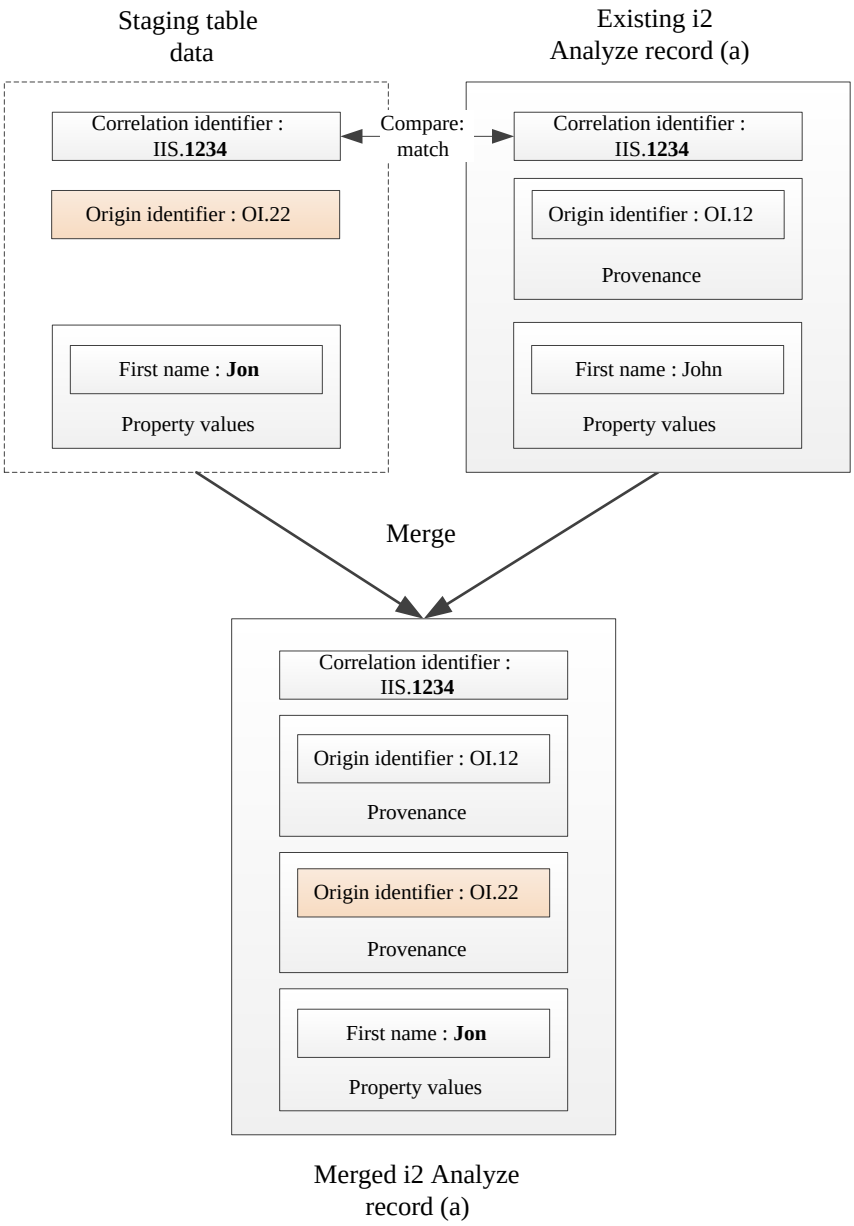
  If the default behavior does not match the requirements of your deployment, you can change the method for defining property values for merged records. For more information, see Define how property values of merged records are calculated.
- If an existing record to be merged contained any notes, the notes are moved to the merged record.
- If an existing record to be merged was an entity record at the end of any links, the links are updated to reference the merged record.

  Note: Any links that were created through Analyst's Notebook are also updated to reference the merged record.

During ingestion, the number of merge operations that occur is reported in the `merge_count` column of the ingestion report.

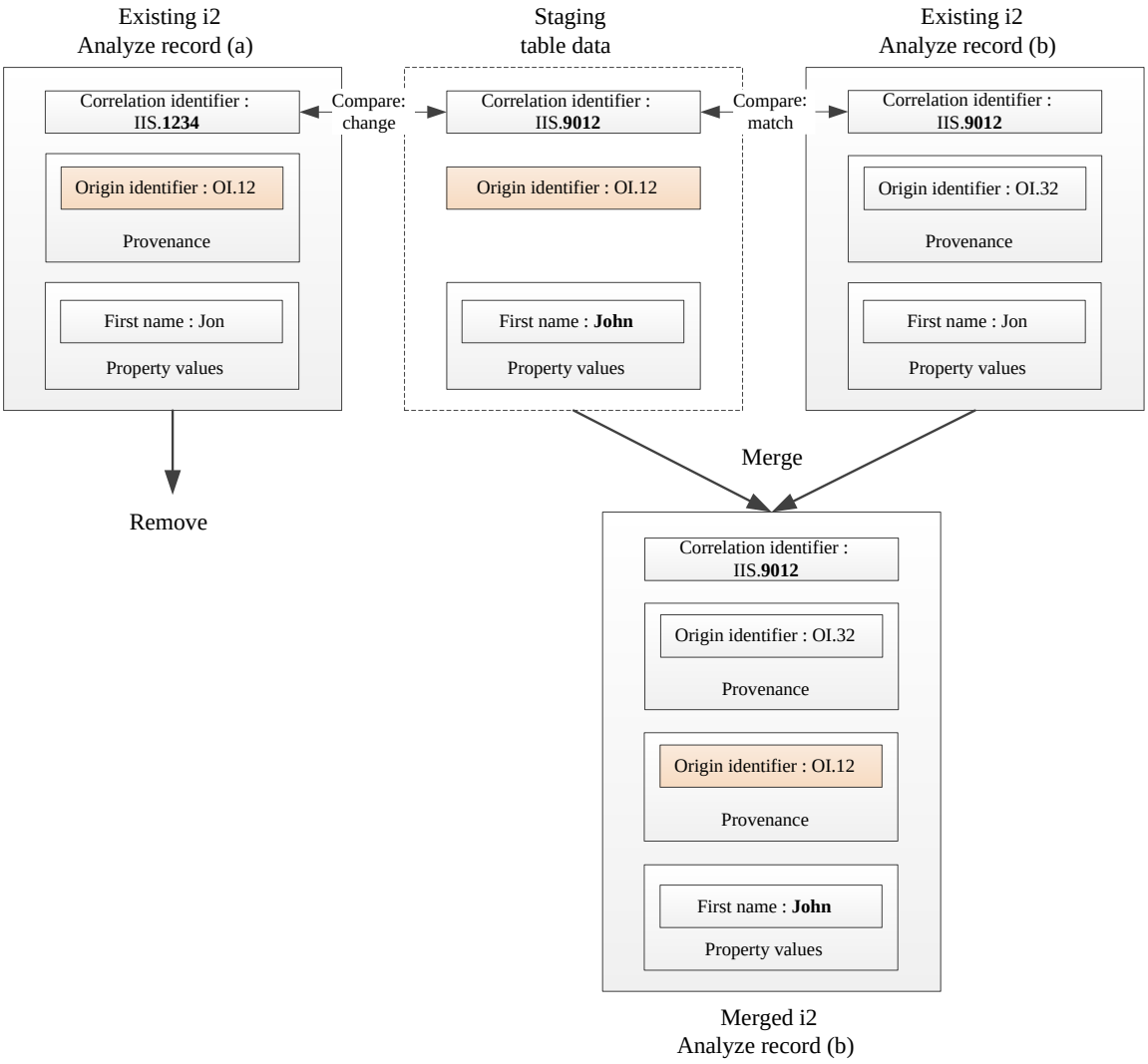
The following diagrams demonstrate the merge operation.

In the first example of a merge operation, data in the staging table is merged into an existing i2 Analyze entity record because the correlation identifiers match and the origin identifiers are different.



In the diagram, the correlation identifiers of data in the staging table and the existing i2 Analyze record (a) match, which causes a merge operation. The existing i2 Analyze record (a) is not associated with the origin identifier of the incoming data. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the data in the staging table row are used. This results in a change to the value for the first name property from "John" to "Jon". The merged i2 Analyze record (a) now contains provenance for the origin identifier `OI.12` and one for the new data, `OI.22`.

In the second example of a merge operation, data in the staging table causes an update to an existing record (a) that changes the correlation identifier to match another record (b) in the Information Store, causing a merge.



In the diagram, the data in the staging table has a different correlation identifier to the record (a) that it is currently associated with by its origin identifier, and the same correlation identifier as another existing record (b). This causes a merge operation. The first existing record (a) no longer has any provenance associated with it, and is removed. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the staging table row are used. This results in a change to the value for the first name property from "Jon" to "John". The merged i2 Analyze record (b) contains multiple pieces of provenance, one for the origin identifier `OI.32` and one for the new data, `OI.12`.

Unmerge

If the data for a merged i2 Analyze record is determined to no longer represent the same real-world object, the i2 Analyze record can be unmerged into two i2 Analyze records. For the Information Store to unmerge records, the correlation identifier or implicit discriminators of the data associated with that record must be changed.

Assuming that the implicit discriminators are compatible, the unmerge operation occurs when the correlation identifier of a row in the staging table is different from the correlation identifier on the merged record that it is currently associated with by its origin identifier.

After the unmerge operation, the following statements are true for the existing i2 Analyze record:

- The piece of provenance for the source information that caused the operation is unmerged from the record.
- The property values for the record are taken from the source information associated with the provenance that has the most recent value for `source_last_updated`.

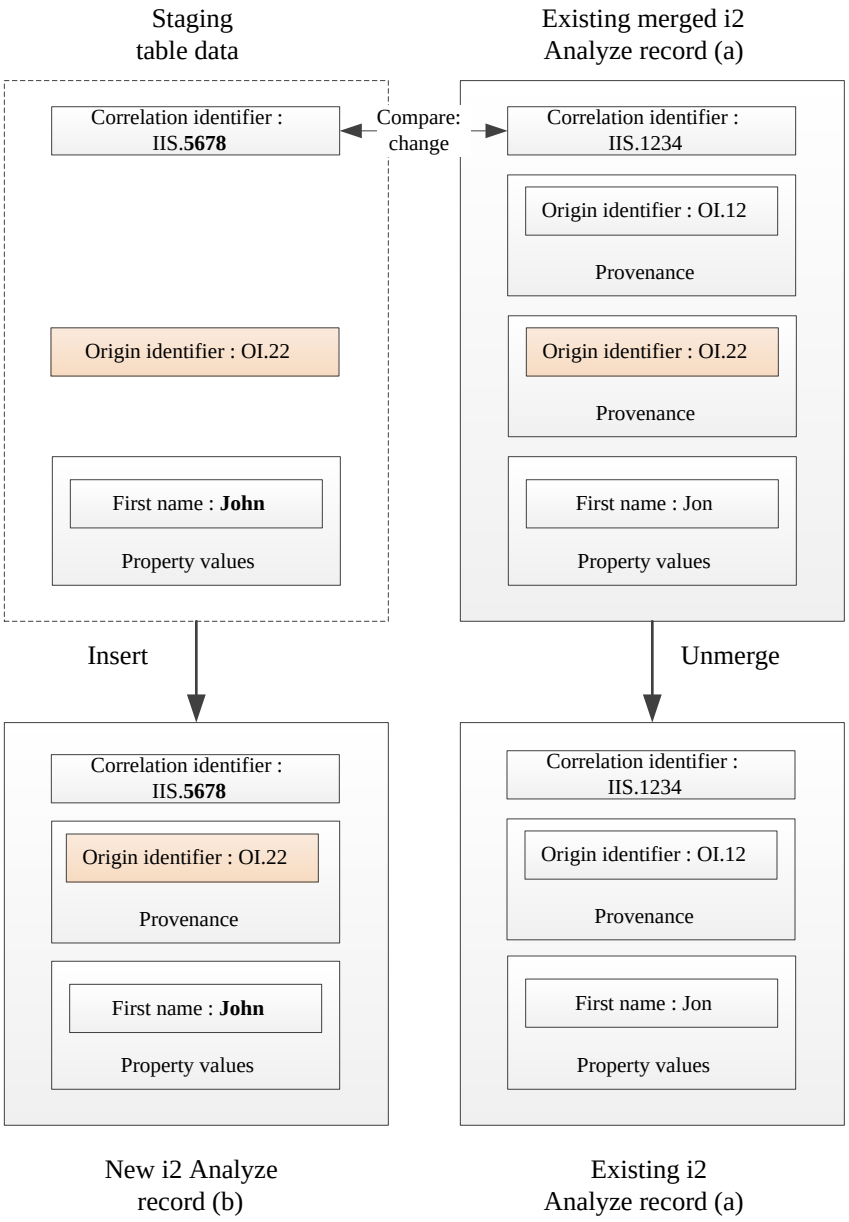
  If only one piece of provenance for a record has a value for the `source_last_updated` column, the property values from the source information that is associated with that provenance are used. Otherwise, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The piece of provenance that is last in the order is chosen. To ensure data consistency, update your existing records with a value for the `source_last_updated` column before you start to use correlation, and continue to update the value.

  If the default behavior does not match the requirements of your deployment, you can change the method for defining property values for merged records. For more information, see Define how property values of merged records are calculated.
- All notes remain on the record.
- The *last updated time* of the record is updated to the time that the unmerge operation occurred.
- If the existing record was an entity record at the end of any links, any links to the unmerged piece of provenance are updated to reference the record that now contains the provenance.

After the unmerge operation, depending on the change in correlation identifier that is presented to the Information Store, either a new i2 Analyze record is inserted or a merge operation is completed. During ingestion, this process is reported in the `unmerge_count`, `insert_count`, and `merge_count` columns of the ingestion report.

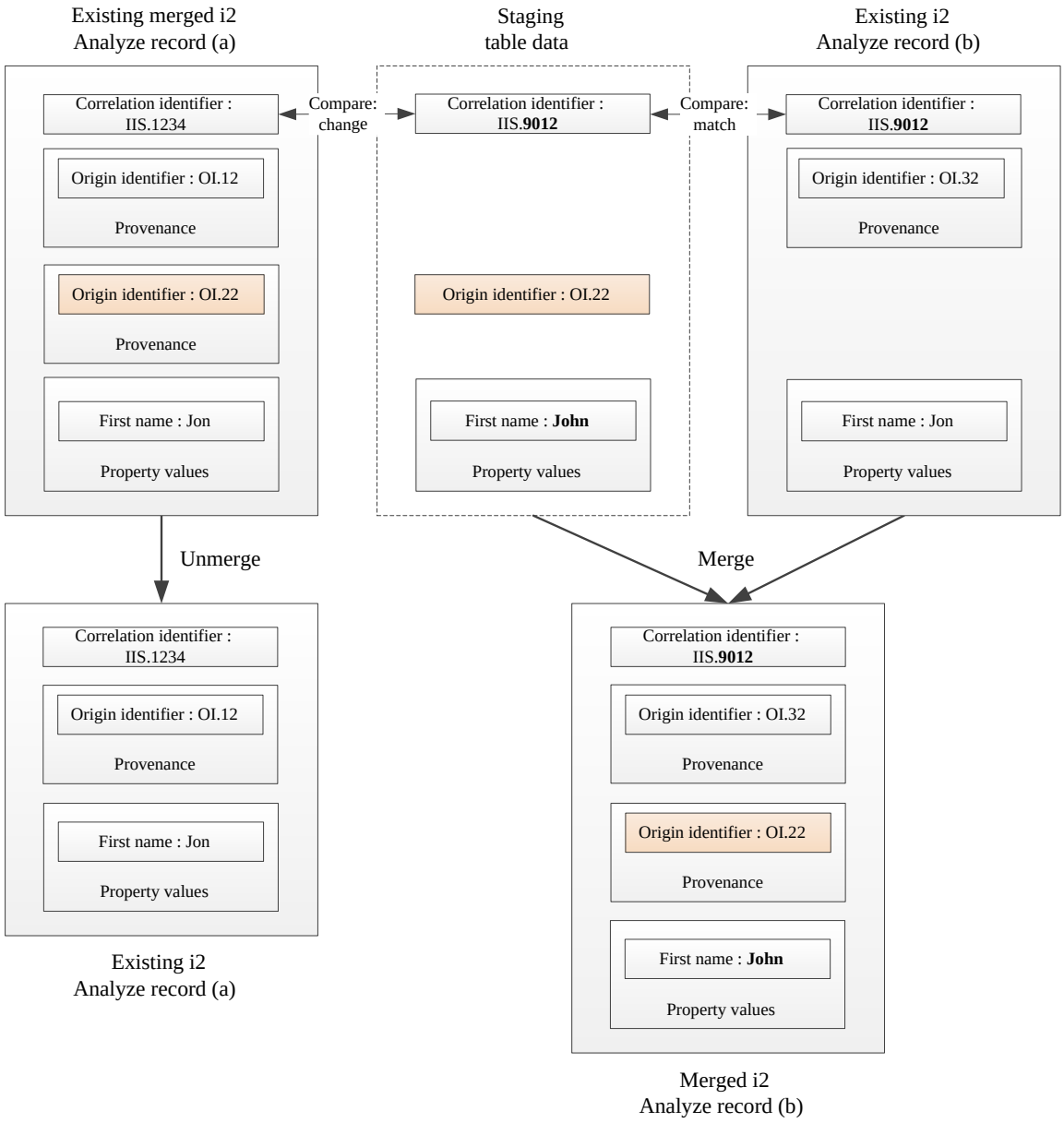
The following diagrams demonstrate the unmerge operation. In each diagram, the data that is ingested from the staging table contains a different correlation identifier to the one on the record that it is associated with by its origin identifier. One unmerge operation results in a new record, and one results in a merge operation.

In the first example of an unmerge, the correlation identifier of the provenance that is unmerged from an existing record (a) does not match with another correlation identifier in the staging table or the Information Store. A new i2 Analyze record (b) is created with the property values from the staging table.



In the diagram, the data in the staging table has a different correlation identifier to the record (a) that it is currently associated with. This causes an unmerge operation. The provenance is unmerged from the existing record (a). The existing record (a) only contains the provenance for the origin identifier `OI.12` and the property values from the source information that is associated with that provenance. The correlation identifier of the staging table data does not match with any others in the Information Store, so a new record (b) is inserted.

In the second example of an unmerge, if the correlation identifier of the provenance that is unmerged from an existing record (a) now matches the correlation identifier of another record (b), a merge operation is performed.



In the diagram, the data in the staging table that is ingested has a different correlation identifier to the record (a) that it is currently associated with. This causes the unmerge operation. The origin identifier and provenance are unmerged from the existing record (a). The existing record (a) now only contains the provenance for the origin identifier `OI.12` and the property values from the source information that is associated with that provenance.

The correlation identifier of the staging table data now matches with another record (b) in the Information Store, so a merge operation occurs. In this example, it is assumed that the staging table data is more recent than the existing data. As part of the merge, the property values from the staging table row are used. This results in a change to the value for the first name property from "Jon" to "John". The merged i2 Analyze record (b) now contains two pieces of provenance, one for the origin identifier `OI.32` and one for the new data, `OI.22`.

For more information about the behavior of a merge operation, see Merge.

Hidden circular links

As a result of a merge operation, circular links might occur in the Information Store. The Information Store does not support circular links.

If data in the staging table causes existing records that are linked to each other in the Information Store to merge, the link becomes circular. As part of this operation, the existing link record is hidden in the Information Store. When a link record is hidden in the Information Store, analysts are unable to discover the link in search results and analysis.

A hidden link is revealed when the merged record at its ends is unmerged. After a hidden link is revealed, analysts are able to discover the link in search results and analysis.

Define how property values of merged records are calculated

In the Information Store, each property of an i2 Analyze record can have only one value. When multiple pieces of source data contribute to an i2 Analyze record, the system must calculate a single value for each property type.

Intended audience

The information about defining the property values of merged records is intended for users who are database administrators and experienced in SQL. To define the property values, you must write complex SQL view definition statements.

Important: You must write and test the SQL view statements for defining the property values of merged records in a non-production deployment of i2 Analyze. If you create an incorrect view, you might have to clear all the data from the system. Before you implement your view definitions in a production system, you must complete extensive testing in your development and test environments.

Why define the property values

When a record contains more than one piece of provenance, it is a merged record. The properties for an i2 Analyze record are calculated when a merge or unmerge operation occurs, or when provenance is removed from a record.

By default, all of the property values for a record come from the source data that contributed to the record with the most recent source-last-updated-time. If no source data has a source-last-updated-time, the property values to use are determined by the ascending order of the origin identifier keys that are associated with the record. The source data that is last in the order is chosen.

If the default source-last-updated-time behavior does not match the requirements of your deployment, you can define how the property values are calculated for merged i2 Analyze records. You might define your own rules when multiple data sources contain values for different properties of an item type or one data source is more reliable for a particular item or property type.

To demonstrate when it is useful to define how to calculate the property values for merged records, imagine that the following two pieces of source data contributed to an i2 Analyze record of type Person:

Origin identifier	Correlation identifier	Ingestion source name	Source last updated	First given name
DVLA1234	II1	DVLA	12:20:22 09/10/2018	John
PNC5678	II1	PNC	14:10:43 09/10/2018	Jon

In the default behavior, the property values are used from the source data with the most recent value for the Source last updated column. The property values from the row with the ingestion source name of `PNC` are used for the merged i2 Analyze record, and the record gets the value of `Jon` for the first given name property.

If you know that data from the `DVLA` ingestion source is more reliable for this item type, you can define that source data with the value of `DVLA` for the ingestion source name takes precedence. By using this definition, the i2 Analyze record gets the value of `John` for the first given name property.

After you define this rule for the Person entity type, all future updates to the records of this item type take the property values from the `DVLA` ingestion source if it is present in any of the source data that contributed to a merged i2 Analyze record.

For more information about how to enable this function, and create your own rules, see Defining the property values of merged i2 Analyze records.

Defining the property values of merged i2 Analyze records

By default, the mechanism for controlling the property values of merged records is not enabled in a deployment of i2 Analyze. To define which property values are used for merged records, you must inform i2 Analyze that you intend to do this and then customize views in the Information Store database.

Before you begin

Important: You must write and test the SQL view statements for defining the property values of merged records in a non-production deployment of i2 Analyze. If you create an incorrect view, you might have to clear all the data from the system. Before you implement your view definitions in a production system, you must complete extensive testing in your development and test environments.

About this task

To inform i2 Analyze that you intend to define the property values of merged records, you must run the `enableMergedPropertyValues` toolkit task. You can take control of the property values for records of specific item types, or all item types in the i2 Analyze schema.

When you run the toolkit task, two views are created for each item type that you specify. The views are created in the `IS_Public` schema, and are named with the display name of each item type. The two views provide the mechanism for viewing the data that contributes to the merged i2 Analyze records, and for creating rules to calculate a single value for each property:

- The view whose name is suffixed with `_MCV` contains all the data that contributed to each merged i2 Analyze record of that item type. This is the merge contributors view. This view is designed for you to inspect the column names and property values from the data in the Information Store. You must not modify the merge contributors view.

  For more information about the merge contributors view, see The merge contributors view.

- The view whose name is suffixed with `_MPVDV` is where you can define the rules for calculating the property values of merged records. This is the merged property values definition view. You can modify this view to define how the merged property values are calculated from the data in the merge contributors view.

  When you generate the merged property values definition view, the default source-last-updated-time behavior is implemented in the view.

  For more information about modifying the definition, and some examples, see The merged property values definition view.

For example, the Person entity type from the example law enforcement schema produces the `IS_Public.E_Person_MCV` and `IS_Public.E_Person_MPVDV` views.

Procedure

1. Identify the item types that you want to define the property values for. For more information about why you might want to do this, see Define how property values of merged records are calculated.
2. Run the `enableMergedPropertyValues` toolkit task for each of the item types that you identified in step 1.
   a. Open a command prompt and navigate to the `toolkit/scripts` directory.
   b. Run the `enableMergedPropertyValues` toolkit task:

      For example, to create the views for the Person entity type from the example law enforcement schema, run the following command:

      ```
      setup -t enableMergedPropertyValues -p schemaTypeId=ET5
      ```

   You can run the toolkit task with `schemaTypeId=`*`ItemTypeId`* or without any arguments to create the views for all item types in the i2 Analyze schema. You can run the task multiple times with different item type identifiers.
3. Inspect the merge contributors view (`_MCV`) that is created in the `IS_Public` schema and identify any rules to create for calculating the property values of merged records.

   For more information about the merge contributors view, see The merge contributors view.
4. Modify the SQL create statement for the merged property values definition view (`_MPVDV`) that is created in the `IS_Public` schema to create a view that implements the rules you identified in step 3.

   For more information about modifying the definition, and some examples, see The merged property values definition view.

What to do next

After you ingest data that causes correlation operations, ensure that the property values for the merged i2 Analyze records match your expectations by searching for them. If you can ingest data successfully, and the property values are correct in your testing, the merged property values definition view is correct. If the property values are not correct in all cases, you must continue to modify the merged property values definition view and test the results until your requirements are met.

It is recommended that you keep a backup of your complete merged property values definition view.

During the testing of your deployment, you might need to clear any test data from the system to ensure that your view is behaving correctly. For more information, see Clearing data from the system.

You must maintain the merged property values definition view for the lifetime of your deployment. If you change the i2 Analyze schema file, you must update the SQL statement for your view to match any changes. For example, if you add a property type to an item type, the new property must be added to the SQL statement for your merged property values definition view. You do not need to update the merge contributors view. The merge contributors view is updated when the schema is updated so that you can inspect the column name for the new property. For more information, see How to maintain your merged property values definition view.

After you test the view definition in your non-production database, you can implement the view on your production database.

The merge contributors view

In the merge contributors view you can see the source data that contributes to merged i2 Analyze records. By inspecting the view, you can gain an understanding about the data in your system and identify any rules to create that change the way that the property values are calculated.

To understand the merge contributors view, you must first understand the identifiers that the Information Store and i2 Analyze records use. For more information about i2 Analyze identifiers, see Identifiers in i2 Analyze records.

The merge contributors view contains the following details about each piece of source data:

- **item_id**

  Item identifiers are used to distinguish i2 Analyze records of a particular type uniquely within the Information Store. Multiple rows in the merge contributors view can have the same item identifier. Each row that has the same item identifier represents one piece of source data that contributed to the i2 Analyze record with that identifier.

  In the merge contributors view, the item identifier is stored in the `item_id` column.

- **record_id**

  Record identifiers distinguish i2 Analyze records uniquely throughout the deployment. In the merge contributors view, there is a one-to-one mapping between record identifiers and item identifiers.

  In the merge contributors view, the record identifier is stored in the `record_id` column.

- **origin_id_type** and **origin_id_keys**

  The origin identifier is used to reference data in its original source. Each piece of source data has a unique origin identifier.

  In the merge contributors view, the origin identifier is stored in the `origin_id_type` and `origin_id_keys` columns.

- **source_last_updated**

  When data is ingested into the Information Store, you can provide values for when the data was last updated in the data source from which it originated. The values in this column are used for the default source-last-updated-time behavior.

  In the merge contributors view, the last updated time is stored in the `source_last_updated` column.

- **Property values**

  There is a column for each property type in the i2 Analyze schema. Each column is the display name of the property type prefixed with `p_`.

For example, values for the First Given Name property type for a Person entity type from the example law enforcement schema are stored in the `p_first_given_name` column.

- **correlation_id_type** and **correlation_id_key**

  The correlation identifier is used to indicate that data is about a specific real-world object. For multiple pieces of data to contribute to the same i2 Analyze record, they must have the same correlation identifier.

  In the merge contributors view, the correlation identifier is stored in the `correlation_id_type` and `correlation_id_key` columns.

- **ingestion_source_name**

  When data is ingested into the Information Store, you must provide a data source name that identifies the data source from which it originated. This is stored as the ingestion source name in the Information Store.

  In the merge contributors view, the ingestion source name is stored in the `ingestion_source_name` column.

In most scenarios, the `origin_id_type`, `origin_id_keys`, `source_last_updated`, and `ingestion_source_name` values can be used for choosing the source data to provide the property values.

The following table shows an example merge contributors view for a Person entity type where two pieces of source data contribute to the same i2 Analyze record:

	Source 1	**Source 2**
record_id	2R1HCAGm2FnmSkdgRbRjpTuW	2R1HCAGm2FnmSkdgRbRjpTuW
item_id	1	1
origin_id_type	DVLA	PNC
origin_id_keys	1234	5678
source_last_updated	12:20:22 09/10/2018	14:10:43 09/10/2018
p_first_give_name	John	Jon
p_middle_name	James	
p_...	...	...
correlation_id_type	II	II
correlation_id_key	1	1
ingestion_source_name	DVLA	PNC

In this example merge contributors view, you can see the following details:

- The data originates from two ingestion sources; DVLA and PNC

- The data from the PNC ingestion source was updated more recently than the data from the DVLA ingestion source
- The data from the PNC ingestion source is missing a value for the middle name property

You might already know some of these details about the data in your system, or you might be able to determine these differences from inspecting the contents of the view. If you decide that you want to change the default behavior for calculating property values, you must customize the merged property values definition view. For more information about how to customize the merged property values definition view, see The merged property values definition view.

The merged property values definition view

The merged property values definition view is used to calculate the property values of merged i2 Analyze records. You can modify the view to define which property values records receive.

The merged property values definition view is used to calculate a single value for each property type for each merged record.

After you inspect the merge contributors view and identify the rules that you want to define, you must modify the SQL statement to create a view that matches your requirements. For more information about the merge contributors view, see The merge contributors view.

After you create a merged property values definition view, you must maintain the view through the lifetime of your deployment. For more information, see How to maintain your merged property values definition view.

Requirements of the view

The merged property values definition view is over the merge contributors view. To produce a single value for each property type, the view must contain one row for each `item_id` and populate each column with a value. You can include NULL values for property types that have no value.

After you customize the merged property values definition view, it must conform to the following requirements:

- The view must have columns for the item identifier and every property type column in the merge contributors view. The column names in both views must be the same.

  The merged property values definition view must not contain the `origin_id_type`, `origin_id_keys`, `source_last_updated`, `correlation_id_type`, `correlation_id_keys`, or `ingestion_source_name` columns.
- Each item identifier must be unique.

  Important: You cannot ingest data into the Information Store if the view contains multiple rows with the same item identifier.

Default behavior

By default, the merged property values definition view uses the default source-last-updated-time behavior to define which source data the property values come from. You can see the SQL statement for the view in IBM Data Studio or SQL Server Management Studio. An example of the SQL statement for the generated view for the Person entity from the example law enforcement schema is:

```
SELECT item_id, p_unique_reference, p_title,
     ... (all property type columns) ...
     p2_issued_date_and_time, p3_issued_date_and_time
FROM
```

```
   (
       SELECT MCV.*,
            ROW_NUMBER ( ) OVER (PARTITION BY MCV.item_id
                                 ORDER BY  source_last_updated DESC NULLS
 LAST,
                                           origin_id_keys DESC,
                                           origin_id_type DESC)
                           AS partition_row_number
       FROM IS_Public.E_Person_MCV AS MCV
       INNER JOIN IS_Public.E_Person_STP AS STP
               ON MCV.item_id = STP.item_id
   ) AS P
WHERE partition_row_number = 1
```

The SQL statement for the default view works in the following way:

- To ensure that the view contains the correct columns, the first `SELECT` statement returns the `item_id` and all property type columns. For example, `p_title`.
- To split the rows into groups that contain only the data that contributed to a single merged record, the `OVER` and `PARTITION BY` clauses are used on the `item_id` column.
- To define the value for each property, the `ROW_NUMBER` function and `ORDER BY` clauses are used. For the default behavior, the `ORDER BY` clause is used on the `source_last_updated`, `origin_id_keys`, and `origin_id_type` columns. Each column is in descending order. By using `NULLS LAST`, you ensure that if the `source_last_updated` column does not contain a value, it is listed last.

  The `ROW_NUMBER` function returns the number of each row in the partition. After the rows are ordered, the first row is number `1`. The initial `SELECT` statement uses a `WHERE` clause to select the values for each row with a value of `1` for `partition_row_number`. Because only one row has a value of `1`, the view contains only one row for each item identifier.
- When you enable merged property views by using the `enableMergedPropertyViews` toolkit task, a table alias for an internal staging table is also created. For example, `IS_Public.E_Person_STP` is the table alias for the staging table that is created during a Person item type ingestion.

  During the ingestion process, the internal staging table contains the item identifiers of the records to be inserted or updated as part of the current ingestion in the `item_id` column. When the property values for records are calculated by using the `_MPVDV` view, it uses the information in the `_STP` alias to restrict the number of rows that are processed. The following statement from the previous example ensures that the `_MPVDV` view uses the alias for a specific item type:

  ```
  INNER JOIN IS_Public.E_Person_STP AS STP
                ON MCV.item_id = STP.item_id
  ```

Examples

You can modify the following example views to match the data in your i2 Analyze schema or to meet your property value requirements:

Ingestion Source Name precedence

In this example, the data is ordered so that a specified ingestion source takes precedence. For example, the `DVLA` ingestion source takes precedence over the `PNC` one.

```
SELECT item_id, p_unique_reference, p_title,
     ... (all property type columns) ...
     p2_issued_date_and_time, p3_issued_date_and_time
FROM  (
       SELECT MCV.*,
```

```
            ROW_NUMBER ( ) OVER (PARTITION BY  MCV.item_id
                                 ORDER BY CASE ingestion_source_name
                                               WHEN 'DVLA' THEN 1
                                               WHEN 'PNC' THEN 2
                                               ELSE 3
                                          END
                                ) AS partition_row_number
        FROM IS_Public.E_Person_MCV AS MCV
        INNER JOIN IS_Public.E_Person_STP AS STP
                ON MCV.item_id = STP.item_id
    ) AS P
WHERE partition_row_number = 1
```

If an ingestion source name does not match `DVLA` or `PNC`, it is ordered last. If multiple contributing pieces of data have the same ingestion source name, the behavior is non-deterministic. In a production system, you must include another clause that provides deterministic ordering in all scenarios.

To order the ingestion sources, the `ORDER BY CASE` clause is used. The rows are ordered by the value that they are assigned in the `CASE` expression. For more information about the `ORDER BY CASE` statement:

- For PostgreSQL, see Sorting Rows (ORDER BY) and Simple CASE.
- For SQL Server, see Using CASE in an ORDER BY clause.
- For Db2, see ORDER BY clause and CASE expression.

The `ROW_NUMBER` function and the `OVER` and `PARTITION BY` clauses are used in the same way as the view for the default behavior.

Source last updated and non-NULL

In this example, the data is ordered by the source last updated time. However if there is a NULL value for a particular property type, the value is taken from the data that has the next most recent time. This process is completed until a value for the property is found, or no data is left for that record. The definitions are slightly different depending on the database management system that hosts the Information Store database.

- **Db2**

```
SELECT DISTINCT
       item_id,

 FIRST_VALUE(p_first_given_name,
             'IGNORE NULLS')
             OVER (  PARTITION BY    MCV.item_id
                     ORDER BY        source_last_updated DESC NULLS LAST
                     RANGE BETWEEN   UNBOUNDED PRECEDING AND
                                     UNBOUNDED FOLLOWING
                  ) AS p_first_given_name,

... (FIRST_VALUE function for all property types) ...

 FIRST_VALUE(CAST(p_additional_informatio AS VARCHAR(1000)),
                  'IGNORE NULLS')
             OVER (  PARTITION BY    MCV.item_id
                     ORDER BY        source_last_updated DESC NULLS LAST
                     RANGE BETWEEN   UNBOUNDED PRECEDING AND
                                     UNBOUNDED FOLLOWING
                  ) AS p_additional_informatio

FROM        IS_Public.E_Person_MCV
```

```
INNER JOIN  IS_Public.E_Person_STP AS STP
                ON MCV.item_id = STP.item_id
```

- **PostgreSQL and SQL Server**

```
SELECT DISTINCT
       P.item_id,
       P1.p_first_given_name,
       P2.p_additional_informatio

    ... (all property type columns) ...

FROM  IS_Public.E_Person_MCV AS P
      LEFT OUTER JOIN
      ( SELECT DISTINCT
               MCV.item_id,
               FIRST_VALUE(p_first_given_name)
                   OVER ( PARTITION BY   MCV.item_id
                          ORDER BY       CASE source_last_updated
                                         WHEN NULL THEN
                                               CAST('0001-01-01' AS date)
                                         ELSE
                                               source_last_updated
                                         END DESC
                          RANGE BETWEEN  UNBOUNDED PRECEDING AND
                                         UNBOUNDED FOLLOWING
                        ) AS p_first_given_name
        FROM       IS_Public.E_Person_MCV AS MCV
        INNER JOIN IS_Public.E_Person_STP AS STP
                ON MCV.item_id = STP.item_id
        WHERE      p_first_given_name IS NOT NULL
      ) AS P1

      ON P1.item_id = P.item_id

      ... (LEFT OUTER JOIN and FIRST_VALUE function for all property
 types) ...

      LEFT OUTER JOIN
      ( SELECT DISTINCT
               MCV.item_id,
               FIRST_VALUE(CAST(p_additional_informatio AS VARCHAR(1000)))
                   OVER ( PARTITION BY   MCV.item_id
                          ORDER BY       CASE source_last_updated
                                         WHEN NULL THEN
                                               CAST('0001-01-01' AS date)
                                         ELSE
                                               source_last_updated
                                         END DESC
                          RANGE BETWEEN  UNBOUNDED PRECEDING AND
                                         UNBOUNDED FOLLOWING
                        ) AS p_additional_informatio
        FROM       IS_Public.E_Person_MCV AS MCV
        INNER JOIN IS_Public.E_Person_STP AS STP
                ON MCV.item_id = STP.item_id
        WHERE      p_additional_informatio IS NOT NULL
      ) AS P2

      ON P2.item_id = P.item_id
```

This example uses the `FIRST_VALUE` function to identify which data has the first non-NULL value for a property type. The `FIRST_VALUE` function returns the first value in an ordered set of values. You must

specify a scalar expression, and `PARTITION BY`, `ORDER BY`, and `RANGE` clauses. In the example, the scalar expressions are the property type columns. The `PARTITION BY` and `ORDER BY` clauses are similar to the other examples, where they act on the item identifier and the source last updated time.

In Db2, the `FIRST_VALUE` function allows you to ignore null values by specifying `'IGNORE NULLS'` in the expression. This functionality is not available in PostgreSQL or SQL Server. To achieve the same result, you must perform a series of `LEFT OUTER JOIN`s for each `FIRST_VALUE` function. In the `ORDER BY CASE` clause for each `FIRST_VALUE` function, when the value for the column in the `FIRST_VALUE` function is `NULL` you can set the source last updated time to `0001-01-01`. Setting the last updated time to this value and ordering the partition this way ensures that the row with a value for the property and the most recent source last updated time is returned.

For more information about the `FIRST_VALUE` function:

- For PostgreSQL, see Window Functions.
- For SQL Server, see FIRST_VALUE (Transact-SQL).
- For Db2, see OLAP specification - FIRST_VALUE.

To use the `FIRST_VALUE` function, you must define a function for each property type separately. In the previous examples, there are two examples of the `FIRST_VALUE` function for two property types:

- The first acts on the `p_first_given_name` column.
- The second acts on the `p_additional_informatio` column. The same process is completed, however in the merge contributors view the `p_additional_informatio` column is of data type CLOB(32M). This data type is not supported by the `FIRST_VALUE` function. To use this column, you must cast it into a supported data type. For example, VARCHAR(1000). You must ensure that if a column is cast, that you do not lose any data.

How to maintain your merged property values definition view

The merged property values definition views include all of the property types for an item type as they were when the view was created. If you change the i2 Analyze schema to add a property type to an item type, the merged property values definition view is now incorrect.

The merged property values definition view does not contain a column for the new property type, and therefore cannot populate an i2 Analyze record with a value for that property. When you add property types to the i2 Analyze schema for an item type, you must update any merged property values definition view to include any new property types. i2 Analyze updates the merge contributors view when the schema is updated, you can use the updated merge contributors view to see the name of the new column.

Note: If you do not update your merged property values definition view, you cannot ingest data into the Information Store.

To stop controlling the property values of merged records for an item type, you can inform i2 Analyze that you no longer intend to use this function. Inform i2 Analyze by running the `disableMergedPropertyValues` toolkit task. You can specify `schemaTypeId=<ItemTypeId>` to remove the views for one item type only, or without any arguments to remove the views for every item type in the i2 Analyze schema. After you run the toolkit task, the default source-last-updated-time behavior is used.

You can modify the merged property values definition view or run the `disableMergedPropertyValues` toolkit task at any time. The property values of i2 Analyze records that are already in the Information Store are not updated. If you ingest any data that updates an existing

i2 Analyze record, the property values for that record are recalculated with the new merged property values definition view.

Ingesting example correlation data

The correlation example provides sets of data that were passed through a matching engine so that each row of data in the set is associated with a correlation identifier. You can ingest the example data, and then inspect the items in the Information Store from Analyst's Notebook to demonstrate the correlation behavior.

About this task

Use the `ingestExampleData` toolkit task to ingest example data sets that demonstrate the correlation behavior in i2 Analyze. For more information about the operations that occur, and how the i2 Analyze records are effected, see Correlation operations.

- **`law-enforcement-data-set-2`**

  This data set contains data with correlation identifiers. After you ingest this data set, the Information Store database contains i2 Analyze records with correlation identifiers. No correlation operations occur when you ingest this data set.
- **`law-enforcement-data-set-2-merge`**

  This data set contains data that causes a number of merge operations to occur with some of the i2 Analyze records that were created from the first data set.
- **`law-enforcement-data-set-2-unmerge`**

  This data set contains data that causes a number of unmerge operations to occur with some of the i2 Analyze records that were merged from the second data set.

You can find the example data sets in the `toolkit\examples\data` directory.

Procedure

1. In a command prompt, navigate to the `toolkit\scripts` directory.
2. Ingest the first example data set.
   a. Run the following command to ingest the first example data set:

      ```
      setup -t ingestExampleData -e law-enforcement-data-set-2
      ```

      The `examples\data\law-enforcement-data-set-2` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables. Each row of data contains a correlation identifier type and a unique key value.

      The Information Store now contains i2 Analyze records with correlation identifiers. However, no correlation operations occurred during the ingestion.
   b. In Analyst's Notebook, search for "Julia Yochum" and add the returned entity to the chart.
3. Ingest the `law-enforcement-data-set-2-merge` data to demonstrate merge operations.
   a. Run the following command to ingest the second example data set:

      ```
      setup -t ingestExampleData -e law-enforcement-data-set-2-merge
      ```

      The `examples\data\law-enforcement-data-set-2-merge` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables. Each row of data contains a correlation identifier type and

key value. Each row contains a unique value for the `source_id`, which represents the origin identifier.

In this scenario, the matching engine identified that some of the data represents the same real-world objects as data in the first data set. As part of this process, the correlation identifiers match with some of the existing data in the database, but the origin identifiers are different. These matches cause a number of merge operations to occur during the ingestion.

For example, you can see on line 2 of the `person.csv` file the correlation identifier key is `person_0`, which matches the correlation identifier of the record for person "Julia Yochum". This match causes a merge between the existing record and the incoming row of data. As part of the merge, the property values from the incoming row of data are used for the record, this results in the full name changing to "Julia Yocham". This is an example of the scenario that is described in the first part of Merge.

In the ingestion reports, you can see the number and type of correlation operations that occurred during the ingestion. For more information about understanding the ingestion reports, see Understanding ingestion reports.

b. In Analyst's Notebook, select the chart item that represents Julia Yochum and click **Get changes**. You can see that the name changes due to the merge operation described previously.

4. Ingest the `law-enforcement-data-set-2-unmerge` data to demonstrate unmerge operations.

- Run the following command to ingest the third example data set:

  ```
  setup -t ingestExampleData -e law-enforcement-data-set-2-unmerge
  ```

  The `examples\data\law-enforcement-data-set-2-unmerge` directory contains a set of CSV files with data that was passed through a matching engine and processed to meet the requirements of the staging tables.

  In this scenario, the matching engine identified that some of the data no longer represents the same real-world objects as it did previously. As part of this process, the correlation identifiers of previously merged data are changed. These changes cause a number of unmerge operations to occur during the ingestion.

  In this example, the correlation identifier of the data that was ingested has changed, but the origin identifier remained the same. For example, on line 2 of the `person.csv` file, the correlation identifier key is now `person_1101` for origin identifier `PER:GEN\1101`. Before, this value was `person_0`. This causes an unmerge operation on the record that the data is currently associated with. This is an example of the scenario that is described in the first part of Unmerge.

  You can see the number of unmerge operations that occurred in the ingestion reports.

a. In Analyst's Notebook, search for "Julia Yocham". There are now two entities for Julia Yocham.

What to do next

After you investigate the correlation behavior, you can clear the example data from your system. For more information, see Clearing data from the system.

Information Store data deletion

i2 Analyze supports several different mechanisms for creating records in the Information Store. It also supports different mechanisms for deleting them. This documentation describes how to delete

records individually, selectively, or in bulk; and it describes the circumstances in which each approach is possible or desirable.

This guide is intended for system implementers and database administrators who want to delete data directly from the Information Store. It applies to data added by ingestion from one or more external data sources, and to data added by users who share information through i2 Analyst's Notebook.

Note: For more information about adding data by ingestion, see Information Store data ingestion. For more information about adding data through i2 Analyst's Notebook, see Uploading records to the Information Store.

The tasks that the following topics describe variously require familiarity with the i2 Analyze data model, your database management systems, SQL queries, and IBM Data Studio or SQL Server Management Studio. Some of the approaches also require specialized authorization with the Information Store.

Overview

To comply with data protection regulations, storage restrictions, or other requirements, you might need to delete data from the Information Store. i2 Analyze provides mechanisms for doing so. The mechanism that you choose depends on how the records were created in the Information Store and your reason for deleting them.

There are two ways to delete records from the Information Store:

- Some of the records in the Information Store are created and uploaded through Analyst's Notebook. If a user deletes a record that they or a colleague uploaded, it becomes unreachable for all users, but remains in the Information Store. You can arrange to delete these *soft-deleted* records permanently, either automatically or on demand.
- For all records in the Information Store, you can write identifying rules that target them for deletion individually or as a group. This approach deletes records no matter how they were originally created or what their current state is.

It is common to all record deletion that deleting an entity record forces i2 Analyze to delete the link records that are attached to that entity. A link must always have two ends. If one end is deleted, the link is automatically deleted as well.

It is also common to both of these ways for deleting records from the Information Store that the procedure is permanent. If you might need to recover the deleted data in future, ensure that you have a backup plan in place before you begin.

Note: The demands of synchronizing with the contents of an external source can also require you to delete data from the Information Store. In that situation, you can reflect deleted data in an external source by using the same ingestion pipeline that you use to reflect creation and modification.

For more information about updating the Information Store in this manner, see Updating the Information Store for deleted data.

Deleting shared records

Analyst's Notebook enables authorized users to create and delete records in the Information Store. However, even authorized users can delete only the records that they or their colleagues create, and the operation is not immediately permanent. To delete records completely, you must interact with the Information Store database.

When a user deletes a record in the Information Store through Analyst's Notebook, the effect for all users is instant. The record is no longer returned in the results of search or expand operations, and copies of the record on other Analyst's Notebook charts become orphaned.

However, by default, the data for the record remains in the database until you decide to delete it permanently. The Information Store has a mechanism for removing these "soft-deleted" records. With IBM Db2 and Microsoft SQL Server, you can start the mechanism on an automatic or a manual basis. With PostgreSQL, you can use the manual mechanism or automate it with a third-party tool.

The mechanism for purging (that is, fully deleting soft-deleted) records uses up to three stored procedures from the `IS_Public` schema of the Information Store database.

- `Purge_Soft_Deleted_Records`

  Parameters: none

  Permanently deletes all soft-deleted records of all types from the Information Store.
- `Set_Purge_Soft_Delete_Schedule` *(Not available with PostgreSQL)*

  Parameters: Schedule

  Sets and activates the schedule on which soft-deleted records are automatically deleted from the Information Store.

  If you are using Db2, the parameter must be a string in UNIX® cron format. For more information, see the UNIX cron format.

  If you are using SQL Server, you must use the arguments and values accepted by the sp_add_jobschedule.
- `Remove_Purge_Soft_Delete_Schedule` *(Not available with PostgreSQL)*

  Parameters: none

  Clears and deactivates the automatic deletion schedule.

On Db2 and SQL Server, the purging mechanism uses the same infrastructure as the mechanism for deleting records by rule. Each manual or automated request to purge soft-deleted records from the Information Store causes a set of deletion jobs to be created. There is one job for each entity type and link type in the i2® Analyze schema, and the Db2® task scheduler or SQL Server Agent runs them at the earliest opportunity.

On PostgreSQL, you need a third-party tool such as pg_cron to automate the process of purging soft-deleted records by executing `SELECT IS_Public.Purge_Soft_Deleted_Records()` on a scheduled basis. To stop the process in such circumstances, just delete the schedule.

i2 Analyze keeps logs of purge operations alongside its logs of deletion-by-rule operations, in the Deletion_By_Rule_Log view. In that view, the rule name for all jobs that are associated with purging soft-deleted records is `PURGE_SOFT_DELETE`.

Purging soft-deleted records

Users who delete records from the Information Store through Analyst's Notebook have no way to recover them afterward, but by default they are not completely removed. As the database administrator, you can decide to leave these soft-deleted records in the database, or to purge them manually or automatically.

Before you begin

Purging soft-deleted records from the Information Store requires the same authorization as deleting records by rule. You must connect to the database as a user with the Authorization to delete by rule. The following instructions assume that you are connected to the database as a user with that role.

In addition, if you want to automate purging, you must be able to schedule jobs to run on the database management system:

- If you are using IBM Db2, the Db2 administrative task scheduler must be enabled.
- If you are using Microsoft SQL Server, the SQL Agent service must be started.
- If you are using PostgreSQL, you must download a third-party tool such as pg_cron.

For more information, see Installing a database.

About this task

When users delete records from the Information Store through Analyst's Notebook, the data remains in the Information Store (but is inaccessible to users) unless you do something to change that. The following procedures describe how to purge soft-deleted records manually, how to automate that process, and how to understand the effect of a purge operation.

Procedure

To perform a one-off, manual purge of all soft-deleted records from the Information Store:

- Run `IS_Public.Purge_Soft_Deleted_Records`.
  - For example, for Db2:

    ```
    CALL IS_Public.Purge_Soft_Deleted_Records;
    ```
  - For example, for SQL Server:

    ```
    EXECUTE "IS_Public".Purge_Soft_Deleted_Records;
    ```
  - For example, for PostgreSQL:

    ```
    SELECT IS_Public.Purge_Soft_Deleted_Records()
    ```

  You can use this approach regardless of whether you also configure automatic purging. The call immediately creates jobs for purging soft-deleted records of every type.

With a Db2 or SQL Server database, you can set up the Information Store so that soft-deleted records are purged automatically, on a schedule:

- Run the `IS_Public.Set_Purge_Soft_Delete_Schedule` stored procedure with the schedule that you want to use.
  - For example, for Db2:

    ```
    CALL IS_Public.Set_Purge_Soft_Delete_Schedule('0 0 * * *');
    ```

    For more information about the format of the schedule, see UNIX cron format.
  - For example, for SQL Server:

    ```
    EXECUTE "IS_Public".Set_Purge_Soft_Delete_Schedule
        @freq_type=4,@freq_interval=1,
        @freq_subday_type=1,@active_start_time = 00000;
    ```

    For more information about the arguments and values for the schedule, see sp_add_jobschedule.

    Note: You must not specify values for the `@job_id`, `@job_name`, `@name`, `@enabled` arguments. i2 Analyze provides the values for these arguments.

  In this example, the Information Store is configured to create the jobs that purge any soft-deleted records every day, at midnight.

To turn off automatic purging, call the `IS_Public.Remove_Purge_Soft_Delete_Schedule` stored procedure.

With a PostgreSQL database, where there is no built-in scheduling agent, you need to set up the soft-deletion process for yourself:

- Download a third-party scheduling agent for PostgreSQL, such as pg_cron.
- Use your chosen tool to make the call to `Purge_Soft_Deleted_Records()` on a scheduled basis.
- To end automatic soft deletion, delete the scheduled function call.

Regardless of whether a purge was started manually or automatically, the effect is always the same: a set of jobs is created to remove soft-deleted records from the Information Store. To inspect the status and outcome of those jobs, you can use a view that the Information Store provides:

- In IBM Data Studio, SQL Server Management Studio, or pgAdmin, open the `IS_Public.Deletion_By_Rule_Log` view and look for blocks of jobs where the `rule_name` is `PURGE_SOFT_DELETE`.

  For more information about the contents of the `IS_Public.Deletion_By_Rule_Log` view, see Deletion_By_Rule_Log view.

Deleting records by rule

If you need to delete records from the Information Store for policy reasons, rather than to reflect delete operations that happened elsewhere, you can use the deletion-by-rule mechanism. You can directly delete any records that you select by specifying their property values and metadata.

Through deletion by rule, you can target records that match specific conditions and automate deletion to occur regularly. Deletion can be applied consistently across all of the records in the Information Store, whether ingested from an external source or created in i2 Analyst's Notebook. You can delete records by their source, their update time, or by any of their property values.

A deletion-by-rule operation has the following components:

- A condition specifies the particular records in the Information Store that you want to delete.
- A rule contains the condition in a form that you can store, together with a name and information about how you want a deletion job to be created.
- A job contains the rule and is placed in a queue to run deletion on the specified records.

For deleting records by rule, i2 Analyze provides the following procedures and views:

- Deletion views are based on the entities and links that are defined in the i2 Analyze schema. These views are a simplified representation of the underlying data structures for composing conditions.
- Procedures/functions are supplied for defining and managing deletion rules. If you want a job to be scheduled to run automatically, you can automate it.
- Procedures/functions are supplied for creating deletion jobs based on existing rules.
- Views provide a list of deletion rules, mappings between views and schema types, and a log of deletion jobs, status, and outcomes.

The procedures and views that are associated with deletion by rule are generated automatically when you deploy an Information Store or when you upgrade from an earlier version.

You can construct an SQL query that identifies the data that is to be deleted from the Information Store. When you verify that the correct data is returned in response to the SQL query, you can use that

condition to create a rule as a basis for a deletion job. You do not need to have a detailed understanding of the database structure. You do need a knowledge of the i2 Analyze schema. For more information, see the i2 Analyze data model documentation.

When you want to run a deletion job only once or irregularly, you can create it on a manual basis. You might also have deletion jobs that you want run regularly on automated basis.

If you are using IBM Db2 or Microsoft SQL Server, you can use their built-in agents to configure the schedule for deletion-by-rule job creation according to your requirements. All automated jobs are created according to the same schedule and placed in a queue of jobs to run in order of their creation time.

- With Db2, use the Db2 administrative task scheduler.
- With SQL Server, you need the SQL Server Agent to be running.

Note: For more information, see Installing a database.

If you are using PostgreSQL, which does not have its own scheduling agent, you'll need to use a third-party tool such as pg_cron to create deletion schedules.

Deleting data by rule

Use an SQL query to identify the data to be deleted, then create and store the deletion rule to identify the corresponding data records for deletion. You can define the rule to specify deletion on an automated schedule or a manual basis.

Before you begin

If you want to apply deletion by rule to a remote Information Store hosted on a Db2, you must catalog the remote database by using the local IBM Db2 client. For more information, see the first note in Configuring remote IBM Db2 database storage.

About this task

By default, deletion-by-rule privileges are granted to the Information Store database administration user. If necessary, the privileges can be granted to other users by giving them the `Deletion_By_Rule` role. For more information, see Authorization to delete by rule. The privileges give you access to all the deletion-by-rule views and procedures, as summarized in the following outline of the main tasks.

CAUTION: Grant the `Deletion_By_Rule` role only to users with sufficient knowledge and authority. Use caution when you complete deletion-by-rule tasks as they constitute a powerful mechanism for deletion of data that might be difficult to recover. Ensure that there is a reliable backup in place before you delete by rule.

Procedure

1. Identify the data to be deleted by composing and running an SQL query to select the data from the deletion view. You can use a different view to see details of the deletion views for your database.
2. Check the SQL results to confirm that the selected data is as you expected. You can use the condition in subsequent steps to identify the particular records in the Information Store that you want to be deleted.
3. Create a deletion rule based on the SQL query you verified at step 2 by using the supplied stored procedure. By default when a rule is created, automated job creation is switched off. You can use another view to see a list of rules and their automation status.

4. Create a deletion job manually by using the supplied stored procedure. The procedure creates a deletion job that contains the rule, which is queued to run.
5. Check the status of the job, and that deletion occurred as you expected, by consulting the `IS_Public.Deletion_By_Rule_Log` view. For more information, see Verifying deletion by rule.
6. **Optional:** Automate deletion by rule by using the supplied stored procedure to include the rule in automated job creation.

Identifying the data

How you compose a rule to delete data from the Information Store varies according to the structure and content of your database. The first step is to understand what you need to delete, and then create an SQL query that selects that data from the deletion views.

About this task

Use the deletion views as a basis to create and validate an SQL query that selects the data you want to delete. For more information, see Deletion views and columns.

You can complete this task in pgAdmin (for PostgreSQL), Microsoft SQL Server Management Studio, or IBM Data Studio.

Procedure

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. By using the data in a deletion view, identify the criteria that you can use to delete specific data.
3. Create your SQL query. For example, based on the deletion view `IS_Public.E_Event_DV`, create a query that returns all event entities where the last update in the source data was more than three years ago:
   - For PostgreSQL:

     ```
     SELECT *
     FROM IS_Public.E_Event_DV
     WHERE source_last_updated < CURRENT_DATE - INTERVAL '3 YEARS'
     ```
   - For SQL Server:

     ```
     SELECT *
     FROM "IS_Public".E_Event_DV
     WHERE source_last_updated < DATE_ADD(yyyy, -3, GETDATE())
     ```
   - For Db2:

     ```
     SELECT *
     FROM IS_Public.E_Event_DV
     WHERE source_last_updated < CURRENT_DATE -3 YEARS
     ```
4. Verify that the data that is returned from your SQL query is the data that you want to delete.

What to do next

Create a deletion rule based on your query. For more information, see Creating a deletion rule.

Creating a deletion rule

Use a valid SQL query as a basis for a deletion rule. Store the corresponding deletion rule for use in the deletion process.

Before you begin

Verify the condition that you need for the deletion rule by creating an SQL query that returns the appropriate data. For more information, see Identifying the data.

Procedure

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. Run the `IS_Public.Create_Deletion_Rule` stored procedure. You must specify the mandatory parameter values for the stored procedure.
   a. To give the rule a name by which it can be identified, enter a unique name in `Rule_Name`. Rule names are not case-sensitive.
   b. To identify the deletion view that the rule targets, enter the name in `View_Name`. Enter the view name without the `IS.Public` schema name prefix.
   c. Enter the `WHERE` clause conditions from your SQL query in `Conditions`. Enter the code that follows the `WHERE` keyword in the SQL query.

   For example, see the sample parameter values in the following table. Mandatory values are marked with an asterisk. | Name | Type | Value | |--------------|--------------|---| | Rule_Name* | VARCHAR(50) | OLD EVENTS | | View_Name* | VARCHAR(128) | E_Event_DV | | Conditions* | VARCHAR(1000) | source_last_updated < CURRENT_DATE -3 YEARS |

   Running this procedure does not run the rule. However, the rule is validated to inform you of any errors. If the rule has no conditions, you receive a warning message to ensure that you are aware of the consequences when you do create a deletion job based on the rule

   CAUTION: When a job is created from a rule that has an empty value for the conditions parameter, it deletes every record in the view.

What to do next

To review the rule, browse the data in the `IS_Public.Deletion_Rules` view. For example, see the following rule details.

rule_name	view_name	conditions	automated
OLD EVENTS	E_Event_DV	source_last_updated < CURRENT_DATE -3 YEARS	N

By default, the rule is set up so that you can use it to create jobs manually. For more information, see Creating a deletion job. You can also configure the rule so that jobs are created automatically, according to a schedule. For more information, see Automating deletion by rule.

You can update a stored rule by a procedure similar to rule creation, or use a procedure to delete a rule. For more information, see Functions and stored procedures.

Creating a deletion job

Creating a rule stores it with a name that you can use to refer to it later. To delete records that match the rule, you must create a deletion job.

Before you begin

Create a deletion rule. For more information, see Creating a deletion rule.

About this task

When you create a job, the rule is placed in the job queue and initially has a status of `Pending`. At 10-second intervals, i2 Analyze checks for deletion jobs to run.

Note: Do not run a deletion job while data is being ingested into the Information Store.

Procedure

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. To create the deletion job, run the `IS_Public.Create_Deletion_Job` stored procedure and enter the name of the rule.

What to do next

You can check the status of the job in the `IS_Public.Deletion_By_Rule_Log` view. For more information, see Verifying deletion by rule.

You can automate the rule for deletion on a schedule. For more information, see Automating deletion by rule.

Verifying deletion by rule

You can verify the outcome of each deletion-by-rule job by consulting the `IS_Public.Deletion_By_Rule_Log` view. The status indicates whether an instance of deletion based on a rule is pending, succeeded, or failed.

Procedure

1. Connect to your database as a user with the `Deletion_By_Rule` role.
2. To verify the status of deletion jobs, browse the data in the `IS_Public.Deletion_By_Rule_Log` view.
3. Find the rule and job that you are interested in. You can use the `rule_name` and `job_creation_time` columns to do find the rule.
4. Verify the status or the outcome of the deletion process in the `status` column.

What to do next

For more information about the contents of the log, see Deletion_By_Rule_Log view.

When the job status is `Succeeded`, you can browse the data in the deletion view for indications that the effect of the job on the database is as expected. Alternatively, you can run the SQL statement and verify that there are no matching records.

When the job fails, the job status indicates an error. For more information, see Troubleshooting.

Automating deletion by rule

By default, new deletion rules are configured so that you can create deletion jobs manually. You can configure each rule so that deletion jobs are created automatically according to a schedule.

Before you begin

Running deletion-by-rule jobs on an automated basis requires the Db2® administrative task scheduler if you are using Db2, or the SQL Server Agent to be running if you are using SQL Server, or a third-party tool such as pg_cron if you are using PostgreSQL. For more information, see Installing a database.

About this task

When you automate a deletion rule, it is included in a deletion job that is created by a scheduled task. i2 Analyze processes in sequence all the jobs that are in the queue whether created manually or automatically.

Note: Do not run a deletion job while data is being ingested into the Information Store. For more information about the deletion-by-rule job schedule, see Changing the automated job creation schedule.

To automate a rule in Db2 or SQL Server, complete the following steps:

Procedure

- Connect to your database as a user with the `Deletion_By_Rule` role.
- Run the `IS_Public.Automate_Deletion_Rule` stored procedure and enter the name of the rule. The `automated` flag for the rule is changed from N to Y.
- To view the rule details and confirm an automated setting, browse the data in the `IS_Public.Deletion_Rules` view.

Alternatively, to automate a rule in PostgreSQL, complete these steps:

- Download a third-party scheduling agent such as pg_cron.
- Use the agent to call the `IS_Public.Automate_Deletion_Rule` *function* on a schedule.

Results

At the time when automated deletion by rule is scheduled, a job is created for each rule that is set to automated. Each job is queued and runs in sequence.

You can check the status of the job in the `IS_Public.Deletion_By_Rule_Log` view. For more information, see Verifying deletion by rule.

What to do next

You can configure the schedule for deletion-by-rule job creation according to your requirements. For more information, see Changing the automated job creation schedule.

To disable automated deletion for a rule, either run the `IS_Public.Disable_Deletion_Rule` procedure and enter the rule name (Db2 and SQL Server), or remove the call to the `IS_Public.Automate_Deletion_Rule` function from the agent (PostgreSQL).

Troubleshooting

When a deletion by rule job is completed, information that indicates success or failure is sent to the `IS_Public.Deletion_By_Rule_Log` view. More details are recorded in the `IS_Public.Ingestion_Deletion_Reports` view.

You can see the status of all deletion-by-rule jobs in the `IS_Public.Deletion_By_Rule_Log` view. For more information, see Verifying deletion by rule.

If the value of `status` is not `Succeeded`, the type of error that is recorded informs your diagnosis and troubleshooting steps. The possible status values, explanations, and steps to take for each value are described as follows.

Pending: status unchanged after a significant amount of time

1. Check other jobs in the `IS_Public.Deletion_By_Rule_Log` view to see whether there is a backlog due to the number and complexity of jobs; or jobs are not being processed as expected and a significant amount of time has elapsed since the most recent job was processed.
2. Check whether the server is running. You might need to restart the server. For more information, see Troubleshooting the deployment process.
3. Check whether the server is connected to the database. There might be a network issue. You can find the Opal server logs here: `i2\i2analyze\deploy\wlp\usr\servers\opal-server\logs`

If you are applying deletion by rule to a remote Information Store on Db2, the remote database must be cataloged by using the local Db2® client. If the configuration files are not already updated, complete the update by using the toolkit. For more information, see Configuring remote IBM Db2 database storage.

If you are using IBM Db2 and none of the previous steps resolve the issue, there might be a problem with the Administrative Task Scheduler or Db2. For more information, see Troubleshooting administrative task scheduler.

Server error

1. To assess the extent of the problem, in the `IS_Public.Deletion_By_Rule_Log` view, check the `reject_count` value to see the number of items that are rejected.
2. Access the `IS_Public.Ingestion_Deletion_Reports` view for more information. A specific deletion job can be identified by the `job_id` value, which is identical to the `job_id` value in the `IS_Public.Deletion_By_Rule_Log` view.

Note: Deletion-by-rule automatically populates the `label` column in the `IS_Public.Ingestion_Deletion_Reports` view with the value `Deletion rule:<`*`Deletion Rule Name`*`>`.

Deletion views and columns

Deploying i2 Analyze automatically generates a set of deletion views in the `IS_Public` schema that is part of the Information Store. Using the views, you can browse the record data and write SQL queries that select records for deletion.

Use the `IS_Public.Deletion_View_Lookup` view to determine the deletion view names in your Information Store. You can also use the view to understand how the deletion views relate to the schema type IDs. The following table contains some examples.

Schema type identifier	Display name	Deletion view name
ET2	Event	E_Event_DV
ET5	Person	E_Person_DV
ET8	Communications device	E_Communications_Device_DV
ET10	Account	E_Account_DV
LCO1	Communication	L_Communication_DV
LAS1	Associate	L_Associate_DV
LAC1	Access To	L_Access_To_DV

There is a deletion view for each record type. Each of these views has a suffix of _DV. Prefixes of E_ and L_ are used for entities and links. In IBM Data Studio or SQL Server Management Studio, you can expand a view to see its columns.

Any columns with a prefix of P_ contain values of properties that are defined in the i2 Analyze schema. The other columns contain metadata values that it might be useful to base rules on. Each view has columns for property values, which vary by item type, and columns for metadata values that are common to all records. The following table contains some examples of property columns.

Deletion view name	Column name	Sample value
E_Person_DV	p_unique_reference	P86539K
	p_date_of_birth	1932-10-14
	p_gender	Male
	...	...
L_Access_To_DV	p_unique_reference	ADC153
	p_type_of_use	Account Holder
	...	...

Metadata columns can be useful for creating deletion rule conditions. All deletion views have a common set of metadata columns as described in the following table.

Column name	Description
record_id	The record identifier that distinguishes the i2 Analyze record uniquely throughout the deployment

Column name	Description
item_id	The item identifier that distinguishes i2 Analyze records of a particular type within the Information Store
last_update_time	The timestamp of the most recent i2 Analyze data ingestion
create_time	The timestamp that i2 Analyze assigned at the point of creation of the record
source_names	The names of the sources of the data
source_created	The timestamp from the external source for the creation of the data
source_last_updated	The timestamp from the external source for the last update of the data

Notes:

- In the Information Store, there is a one-to-one mapping between record identifiers and item identifiers. For more information about the identifiers that are used in i2 Analyze, see Identifiers in i2 Analyze records.
- Records that are created through i2 Analyst's Notebook have a value of `ANALYST` in the `source_names` column and null values for `source_created` and `source_last_updated`.

Views that represent link types have a few extra metadata columns as described in the following table.

Column name	Description
from_record_id, from_item_id	The identifiers of the record that constitutes the start of a link
to_record_id, to_item_id	The identifiers of the record that constitutes the end of a link
from_item_type_id	The identifier of the type of the record that constitutes the start of a link
to_item_type_id	The identifier of the type of the record that constitutes the end of a link

Each deletion view has a security dimension column, which contains the security dimension values for each record. For more information, see Security model. It can be useful to delete records based on their security dimension values.

Note: The deletion of data is not subject to restriction based on the i2 Analyze security dimension values, but you can filter data for deletion based on these values.

Functions and stored procedures

The i2 Analyze deletion-by-rule functions depend upon a set of standard functions and stored procedures. You can use these routines to manage deletion rules and create deletion jobs.

The available functions and procedures are described below. All parameters are mandatory.

Note: DB2 and SQL Server use stored procedures that you execute with `CALL` or `EXEC`. PostgreSQL uses functions that you execute with `SELECT`.

- `IS_Public.Automate_Deletion_Rule`

  Parameters: RULE_NAME

  Sets `automated` to `Y` for the named rule to be included in the scheduled automatic creation of deletion jobs.

  Deletion-by-rule jobs are placed in the queue of deletion jobs to be run in sequence.
- `IS_Public.Create_Deletion_Rule`

  Parameters: RULE_NAME, VIEW_NAME, CONDITIONS

  Converts the SQL conditions and named view to a rule that is ready for deletion job creation.

  CAUTION: When a job is created from a rule that has an empty value for the conditions parameter, it deletes every record in the view.
- `IS_Public.Delete_Deletion_Rule`

  Parameters: RULE_NAME

  Deletes the named rule.
- `IS_Public.Disable_Deletion_Rule`

  Parameters: RULE_NAME

  Sets the `automated` parameter to `N` for the rule to be excluded from the scheduled automatic creation of deletion jobs.

  The rule can be used for manual deletion only.
- `IS_Public.Create_Deletion_Job`

  Parameters: RULE_NAME

  Creates a deletion job that is based on the named rule. This procedure can be run for the rule when the value of `automated` is `Y` or `N`. The deletion-by-rule job is placed in the queue of deletion jobs to be run.
- `IS_Public.Update_Deletion_Rule`

  Parameters: RULE_NAME, VIEW_NAME, CONDITIONS

  Updates the named rule by revising the original `VIEW_NAME` or `CONDITIONS`.
- `IS_Public.Update_Deletion_Schedule` *(Not available with PostgreSQL)*

  Parameters: SCHEDULE

  Updates the job creation schedule. For more information about the schedule format, see Changing the automated job creation schedule.
- `IS_Public.Validate_Deletion_Rule`

  Parameters: RULE_NAME

Reports on the SQL Results tab any SQL errors that are found associated with the SQL conditions of the named rule.

The report contains results by `MESSAGE`, `SQLCODE`, and `SQLSTATE` values.

`Deletion_By_Rule_Log` view

The `IS_Public.Deletion_By_Rule_Log` view provides information on the status of the deletion jobs in the queue in relation to each rule. The view also contains details on the results of each deletion job that is run.

You can access the view to check the status of a deletion-by-rule job. For more information, see Verifying deletion by rule. After the deletion job is run, the `IS_Public.Deletion_By_Rule_Log` view contains one or more entries per job that describe the result for each item type. For more information, see Sample use cases.

The information that is contained in the `IS_Public.Deletion_By_Rule_Log` view is described in the following table.

Column name	Description
`status`	Status of the deletion by rule job, which can be Succeeded, Pending, Server error, or Validation error.
`rule_name`	Name of the deletion rule.
`view_name`	Name of the view that the rule is targeting.
`conditions`	Deletion conditions that are applied by the rule.
`job_id`	Identifier for the job, which is also used to identify the job in the `job_id` column of the `Ingestion_Deletion_Reports` view.
`job_creation_time`	Time stamp that indicates when the deletion by rule job was created.
`job_run_time`	Time stamp that indicates when the deletion by rule job was run and corresponds to the value in the `start_time_stamp` column of the `Ingestion_Deletion_Reports` view. There is no value in the `job_run_time` column while the job is pending.
`item_type`	Item type of the records that were deleted. There can be more than one entry in this view for the same job. For example, deleting entity records can also cause link records to be deleted.
`delete_count`	The number of records that are deleted.

Column name	Description
`reject_count`	The number of records that are rejected.

Example of job results logged

See the following tables for an example of how the `IS_Public.Deletion_By_Rule_Log` view might look with completed job details.

status	rule_name	view_name	conditions
Succeeded	OLD ACCOUNTS	E_Account_DV	p_date_opened < CURRENT_DATE - 50 YEARS
Succeeded	OLD ACCOUNTS	E_Account_DV	p_date_opened < CURRENT_DATE - 50 YEARS

job_id	job_creation_time	job_run_time	item_type	delete_count	reject_count
77	2017-12-12 14:29:11.351	2017-12-12 14:29:16.467	LAC1	3	0
77	2017-12-12 14:29:11.351	2017-12-12 14:29:16.467	LTR1	3	0

In this example, deleting three Account entity records caused the deletion in the same job of three Transaction link records that were attached to the Accounts.

Changing the automated job creation schedule

When the database management system that hosts the Information Store is IBM Db2 or Microsoft SQL Server, i2 Analyze creates jobs from all automated deletion rules according to the same schedule. When it's PostgreSQL, you must arrange scheduled job creation for yourself. In all cases, i2 Analyze provides tooling to help.

Db2 and SQL Server

Depending on the database management system that hosts the Information Store database, the IBM Db2 administrative task scheduler or Microsoft SQL Server Agent runs the task that creates deletion-by-rule jobs at intervals that you can configure.

The setting that controls how often the deletion-by-rule jobs are created is stored in the Information Store. Whenever the setting matches the current time, date, and day, the scheduler creates jobs from the deletion rules that have the value of `automated` set to `Y`.

When you deploy i2 Analyze, the setting receives its default value that means that the scheduler looks for jobs to create every hour, on the hour.

- For Db2, a value of `'0 * * * *'` is used.

For more information about the format, see UNIX cron format.

- For SQL Server, the following arguments and values are used:
  `@freq_type=4,@freq_interval=1,@freq_subday_type=8,@freq_subday_interval=1`

  For more information about the arguments and values, see sp_add_jobschedule.

  Note: You must not specify values for the `@job_id`, `@job_name`, `@name`, `@enabled` arguments. i2 Analyze provides the values for these arguments.

To change the schedule for automated deletion, run the `IS_Public.Update_Deletion_Schedule` stored procedure. For example, to run the deletion-by-rule job creation task only at midnight, run the stored procedure in the following ways:

If you are using Db2, you must specify a UNIX cron value for the schedule. For example, run the stored procedure with the following value:

```
CALL IS_Public.Update_Deletion_Schedule('0 0 * * *');
```

Note: The Db2 administrative task scheduler checks for new or updated tasks at 5-minute intervals. If you change the setting, be aware of a potential delay of 5 minutes before it takes effect. You can still create a job from the rule manually if you need to.

If you are using SQL Server, you must specify arguments for the `sp_add_jobschedule` stored procedure. For example, run the stored procedure with the following value:

```
EXECUTE "IS_Public".Update_Deletion_Schedule
    @freq_type=4,@freq_interval=1,
    @freq_subday_type=8,@freq_subday_interval=24;
```

Note: If you are using Db2 and you follow this procedure in a deployment that provides high availability, you must run the stored procedure on the primary and any standby database instances.

PostgreSQL

Unlike Db2 and SQL Server, PostgreSQL does not have a built-in scheduling agent, and so the stored procedures for scheduling are not available in this environment.

To automate and schedule job creation, you must use a third-party agent such as pg_cron to execute `SELECT IS_Public.Create_Deletion_Jobs()` on a scheduled basis.

Note: Try to arrange for the function to run during quiet periods when analysts are generally not using the system, such as overnight or at weekends.

Authorization to delete by rule

The deletion-by-rule capability is enabled by the i2 Analyze Information Store deployment. By default the database user who deploys the Information Store is authorized to delete by rule.

Authorization to delete by rule includes permission to access and run all of the views and procedures for doing so in the `IS_Public` schema. For more information, see Deletion views and columns and Functions and stored procedures. If you do not have this authorization, you can see but not access or run these views and procedures.

The authorization to delete by rule can be granted to other users. You can create a new database user and assign the `Deletion_By_Rule` role to that user, or you can assign the role to an existing user.

For example, you can run the following statement for Db2:

```
GRANT ROLE Deletion_By_Rule TO USER deletion_user
```

Or the following statement for SQL Server:

```
ALTER SERVER ROLE Deletion_By_Rule ADD MEMBER deletion_user
```

Or this statement for PostgreSQL:

```
GRANT Deletion_By_Rule TO deletion_user
```

CAUTION: Assign the `Deletion_By_Rule` role only to users with sufficient knowledge and authority. Exercise caution when you create deletion rules as they constitute a powerful mechanism for deletion of data that cannot be recovered afterward.

Sample use cases

Stored data might be monitored for compliance with international standards, corporate regulations, or storage volume restrictions. Depending on the use case, different rules can be created to meet many different kinds of requirement.

Reasons for data deletion often fall into one or more of the following categories:

- Data is out of date
- Data is obsolete or irrelevant in the context of the purpose of the database
- Data has data privacy or security issues
- Data reaches a storage volume threshold

The SQL statements in the following examples contain the kinds of conditions that are commonly used as a basis for deletion.

Note: If you are using Db2, when you are creating SQL for time-based conditions you can use the support for date and time arithmetic. PostgreSQL provides similar Date/Time Functions and Operators.

Selecting by property value

Select records of all persons who are below a specific age. The rule matches records of persons who are younger than 18 years. This example is one that you might use to delete data that does not suit the specific purpose of the database.

- For Db2:

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   p_date_of_birth > CURRENT_DATE - 18 years
```

- For SQL Server:

```
SELECT  *
FROM    "IS_Public".E_Person_DV
WHERE   p_date_of_birth > DATEADD(yyyy, -18, GETDATE())
```

- For PostgreSQL:

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   p_date_of_birth > CURRENT_DATE - INTERVAL '18 YEARS'
```

Metadata: data source

Select all records that are sourced from the "DMV" data source.

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   source_names LIKE '%DMV%'
```

Metadata: creation date

Use SQL to select any record with a creation date that is not in the current calendar year.

- For Db2:

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   YEAR(create_time) < YEAR(CURRENT_DATE)
```

- For SQL Server:

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   create_time < DATEFROMPARTS(DATEPART(yyyy, GETDATE()), 1, 1)
```

- For PostgreSQL:

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   DATE_PART('YEAR', create_time) < DATE_PART('YEAR', CURRENT_DATE)
```

Metadata: old data

Select communication records older than three months.

- For Db2:

```
SELECT  *
FROM    IS_Public.L_Communication_DV
WHERE   create_time < CURRENT_DATE - 3 MONTHS
```

- For SQL Server:

```
SELECT  *
FROM    IS_Public.L_Communication_DV
WHERE   create_time < DATEADD(mm, -3, GETDATE())
```

- For PostgreSQL:

```
SELECT  *
FROM    IS_Public.L_Communication_DV
WHERE   create_time < CURRENT_DATE - INTERVAL '3 MONTHS'
```

Without the specified link

This case is an example where you require deletion of entities that are not linked to other entities. You might want this type of deletion as a general database cleanup, or to target links of a specific type, for example those granting access to a building or an account. Select records of people who do not have any access links.

The rule matches records of people when the `item_id` is not contained in a `from_item_id` column in the link deletion view. When your rule joins to another deletion view, it is recommended that you identify records by using the `item_id` columns.

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   item_id NOT IN (
            SELECT  from_item_id
            FROM    IS_Public.L_Access_To_DV
        )
```

With the specified links

Select records of any person who is associated with a person named 'Michael Wilson'.

When your rule joins to another deletion view, it is recommended that you identify records by using the `item_id` columns.

```
SELECT  *
FROM    IS_Public.E_Person_DV
WHERE   item_id IN (
            SELECT  A.from_item_id
            FROM    IS_Public.L_Associate_DV AS A
                    INNER JOIN IS_Public.E_Person_DV AS P
                        ON P.item_id = A.to_item_id
            WHERE   P.p_first_given_name = 'Michael' AND
                    P.p_family_name = 'Wilson'
        )
```

Data source, property, and timeframe

This case is an example of the kind of deletion that might be used to clean up a database or reduce storage requirements. Select records from a specific source based on when the last update occurred. The rule matches records of people with the named data source and an update date that is older than three years from the current date.

- For Db2:

  ```
  SELECT  *
  FROM    IS_Public.E_Person_DV
  WHERE   source_names LIKE '%LawEnforcementDB%' AND
          source_last_updated < CURRENT_DATE - 3 YEARS
  ```

- For SQL Server:

  ```
  SELECT  *
  FROM    IS_Public.E_Person_DV
  WHERE   source_names LIKE '%LawEnforcementDB%' AND
          source_last_updated < DATEADD(yyyy, -3, GETDATE())
  ```

- For PostgreSQL:

  ```
  SELECT  *
  FROM    IS_Public.E_Person_DV
  WHERE   source_names LIKE '%LawEnforcementDB%' AND
          source_last_updated < CURRENT_DATE - INTERVAL '3 YEARS'
  ```

Retrieving chart metadata

i2 Analyze enables database administrators and third-party extensions to retrieve information about charts that users upload to the Information Store or the Chart Store. Specifically, you can find out which users have uploaded a particular chart, and when that chart was last accessed.

Note: In this context, "accessed" refers to when the chart was uploaded, downloaded, or had its details (that is, its history) requested by a user.

With this knowledge, you might implement a policy such that charts that haven't been accessed for a certain period of time are automatically deleted - but only after you send an alert to all the users who uploaded the chart to warn them.

Determining when a chart was last accessed

To determine when a chart was last accessed, you can query the chart deletion view, whose name has the following format: `IS_Public.<ChartTableName>_DV`. Provided that the item type in the i2 Analyze schema that represents charts has not been modified, its name is `IS_Public.E_Analyst_S_Notebook_Ch_DV`.

This view has the same structure as other deletion views, but with two extra columns:

Column name	Description
`chart_type`	The type of the chart - that is, web or desktop
`last_seen_time`	The (UTC) date and time when the chart was last accessed

For this purpose, the `last_seen_time` column is pertinent. To determine which charts have not been accessed for approximately 12 months, you might execute the following (Db2) SQL statement:

```
SELECT item_id
FROM IS_Public.E_Analyst_S_Notebook_Ch_DV
WHERE last_seen_time < NOW() - 12 MONTHS;
```

You can then use the list of identifiers that this statement returns to find out which users need to be notified.

Determining which users uploaded a chart

The Information Store database provides a view that contains information about which users have uploaded a version of a particular chart. The default name of this view is `IS_Public.E_Analyst_S_Notebook_Ch_Upload_Users`. It has the following columns:

Column name	Description
`item_id`	The item identifier of the chart record
`user_principal_name`	The name of a user who uploaded any version of the chart

Therefore, to determine which users have uploaded (any version of) a chart that has an `item_id` of `1234`, you might execute the following SQL statement:

```
SELECT user_principal_name
FROM IS_Public.E_Analyst_S_Notebook_Ch_Upload_Users
WHERE item_id = 1234;
```

Note: Even if the same user has uploaded multiple versions of the same chart, their principal name is only returned once for each chart.

With this list of user names in hand, you might decide to use one of the i2 Analyze APIs for sending alerts to users.

Sending alerts to i2 Analyze users

i2 Analyze enables system administrators and third-party extensions to send alerts to users of the system. These alerts appear in the Analyst's Notebook desktop client alongside the alerts from Visual Queries that users are familiar with.

In principle, alerts can contain any information that you think it might be useful for i2 Analyze users to receive. In practice, alerts are often about data in the system - to inform users about a change, for example, or to warn them that something temporary is soon to be deleted.

Important: Alerting is always enabled in deployments that include the Information Store, but by default it is *not* enabled in deployments that include only the Chart Store. To enable alerting in a Chart Store deployment, you must set `EnableAlerting=true` in `DiscoServerSettingsCommon.properties`, and then restart the i2 Analyze server.

Creating alerts

The alerts that you create contain an icon, a title, and an optional message. In addition, they can contain either a group of records or a link to an external resource. Alerts can be created through three different APIs:

- Database stored procedures
- The i2 Analyze Java API
- The i2 Analyze REST API

Each API provides the same control over the contents of alerts. The Java and REST APIs provide additional functionality for specifying the recipients of alerts, based on group membership or command access control permissions. (The stored procedures require recipients to be specified explicitly, one at a time.)

Note: Alerts can only be sent to users who have previously logged in to i2 Analyze. An authorized user who has never logged in will not see alerts that were created and sent before they do so.

i2 Analyze Developer Essentials contains examples of using the Java API and the REST API to create alerts. For examples of using the database stored procedures, keep reading.

Alert structure

The structure of an alert is the same no matter which API creates it. Alerts can contain the following elements:

- **title**

  The title is mandatory and should contain a short string that will be displayed for the alert.
- **message**

  The message is optional. If supplied, it can contain a longer string with more information than the title.
- **icon**

  The icon is optional and determines the image that is shown next to the alert in the user interface. If supplied, it must be one of the following values. If not supplied, it defaults to `INFORMATION`.

  - `SUCCESS`
  - `INFORMATION`
  - `WARNING`

- `ERROR`
- `PLUS`
- `MINUS`

- **record identifiers**

  Record identifiers are optional. If they are supplied, the alert links to the specified records or charts and displays them in a results grid (in the same way that Visual Query alert results are displayed).

  Record identifiers cannot be supplied if an href is supplied.

- **href**

  The href is optional. If supplied, it should contain a URL for a resource that provides more information or context for the alert. The URL is displayed as a hyperlink in the user interface.

  An href cannot be supplied if record identifiers are supplied.

Database API

The Information Store (or Chart Store, when alerting is enabled) database contains two public stored procedures for creating alerts. To create an alert with an optional hyperlink, use `IS_Public.Create_Alert`, which takes the following parameters:

Parameter	Alert element	Db2 type	PostgresSQL type	SQL Server type	Size
`user_principal_name`	n/a	VARCHAR	VARCHAR	NVARCHAR	256
`alert_title`	title	VARGRAPHIC	VARCHAR	NVARCHAR	100
`alert_message`	message	VARGRAPHIC	VARCHAR	NVARCHAR	1000
`alert_link`	href	VARGRAPHIC	VARCHAR	VARCHAR	256
`alert_icon`	icon	VARCHAR	VARCHAR	VARCHAR	20

Note: Since version 4.4.4 of i2 Analyze, the `user_principal_name` parameter actually accepts the user *identifier*. In most deployments, the principal name and the identifier are the same - but if they're not, you must use the identifier instead of the principal name.

For example:

```
[Db2]
CALL IS_Public.Create_Alert('test-user', 'For your information', 'Take a
 look at this link', 'https://example.domain/news/123', 'INFORMATION')

[PostgreSQL]
CALL IS_Public.Create_Alert('test-user', 'For your information', 'Take a
 look at this link', 'https://example.domain/news/123', 'INFORMATION')

[SQLServer]
EXEC IS_Public.Create_Alert 'test-user', 'For your information', 'Take a
 look at this link', 'https://example.domain/news/123', 'INFORMATION'
```

Note: To send the same alert to multiple users, you must execute the stored procedure multiple times, changing the `user_principal_name` as required.

To create an alert with one or more record identifiers, use `IS_Public.Create_Records_Alert`, which takes the following parameters:

Parameter	Alert element	Db2 type	PostgreSQL type	SQL Server type	Size
`user_principal_name`	n/a	VARCHAR	VARCHAR	NVARCHAR	256
`alert_title`	title	VARGRAPHIC	VARCHAR	NVARCHAR	100
`alert_record_ids_csv`	record identifiers	VARCHAR	TEXT	VARCHAR	8000
`alert_message`	message	VARGRAPHIC	VARCHAR	NVARCHAR	1000
`alert_icon`	icon	VARCHAR	VARCHAR	VARCHAR	20

For example:

```
[Db2]
CALL IS_Public.Create_Records_Alert('test-user', 'Records added!',
 'h6cBtBz6CuYEU3YwzqUuZEmrhJ,TaUjsCpKUnmY85YWgBQbeNqou2', 'New records added
 from example source', 'PLUS')

[PostgreSQL]
CALL IS_Public.Create_Records_Alert('test-user', 'Records added!',
 'h6cBtBz6CuYEU3YwzqUuZEmrhJ,TaUjsCpKUnmY85YWgBQbeNqou2', 'New records added
 from example source', 'PLUS')

[SQLServer]
EXEC IS_Public.Create_Records_Alert 'test-user', 'Records added!',
 'h6cBtBz6CuYEU3YwzqUuZEmrhJ,TaUjsCpKUnmY85YWgBQbeNqou2', 'New records added
 from example source', 'PLUS'
```

Note: For both Db2 and SQL Server, any alerts that you create through this stored procedure can contain at most 242 records, because of the sizes of record identifiers and the `alert_record_ids_csv` parameter.

The Java and REST APIs call the stored procedure, so they have the same limit. PostgreSQL does not have the same restriction.

Deleting alerts

To delete an alert programmatically, use the database view named `IS_Public.Alerts_DV`. Alerts can be deleted through this view with standard SQL expressions.

For example, to delete all alerts, execute:

```
DELETE FROM IS_Public.Alerts_DV;
```

To delete specific alerts, use a `WHERE` clause. For example, to delete all alerts sent more than 30 days ago (on Db2):

```
DELETE FROM IS_Public.Alerts_DV WHERE create_time_stamp < NOW() - 30 DAYS;

[PostgreSQL]
DELETE FROM IS_Public.Alerts_DV WHERE create_time_stamp < NOW() - INTERVAL
 '30 DAYS';
```

Note: A user who has received an alert can delete it for themselves through the user interface in i2 Analyst's Notebook.

Scheduling custom server tasks

i2 Analyze supports the development of custom *tasks* that the server executes on a user-defined schedule. A task is simply a Java class that implements a specific interface and performs a custom action when invoked.

This topic describes how to deploy a custom task that you or someone else already created. For information about how to develop a custom task, see the scheduled task example in i2 Analyze Developer Essentials.

Deploying a custom task

To deploy a custom task, the Java class (and any supporting classes) must be packaged in a `.jar` file. You must then copy this `.jar` file to the configuration part of the toolkit. Place it the `WEB-INF/lib` sub-folder of the `opal-services` fragment, in the `toolkit/configuration/fragments` folder.

Note: A new fragment specifically for custom tasks is also an option, provided that it is also added to the `topology.xml` file for the deployment.

Configuration

In order for the i2 Analyze server to discover the custom task, entries that identify it must be present in one of the settings files. A typical location for these entries is the `toolkit/configuration/fragments/opal-services/WEB-INF/classes/DiscoServerSettingsCommon.properties` file.

The new entries have two parts, in the following format:

```
CustomTaskScheduler.<TaskName>.Class=<ImplementationClass>
CustomTaskScheduler.<TaskName>.Expression=<CronExpression>
```

Here, `<TaskName>` is a name that identifies a particular task instance; `<ImplementationClass>` is the fully qualified class name for the implementation of that task; and `<CronExpression>` is a cron schedule expression for when the task should run.

The format of the cron expression is:

```
<minute> <hour> <day-of-month> <month> <day-of-week>
```

For example, a cron expression of `0 0 * * *` results in a task running at midnight every day. See the UNIX cron format documentation for more information.

The following settings define two different tasks that use two different classes, on two different schedules:

```
CustomTaskScheduler.MyTask1.Class=com.example.alert.ScheduledTask
CustomTaskScheduler.MyTask1.Expression=0 0 * * *

CustomTaskScheduler.MyTask2.Class=com.example.cleaner.CleanupTask
CustomTaskScheduler.MyTask2.Expression=0 */4 * * *
```

Deployment

When the `.jar` file is copied into the toolkit, and the `DiscoServerSettingsCommon.properties` file is updated with the custom task class name and the cron schedule, the task is ready for deployment to the i2 Analyze server.

To deploy the custom task to the i2 Analyze server, run `setup -t deployLiberty` to update the configuration with your changes. Then, to set up the task to run according to its defined schedule, run `setup -t startLiberty`.

Monitoring a custom task

The scheduler log stores information about the state of all configured tasks, including when a task was last run, when it will next run, whether it has failed, and so on.

The scheduler log can be found in the `i2_Scheduler.log` file, inside the i2 Analyze server.

Note: If a custom task is configured to use the scheduler tracer (`IScheduledTaskObjects.getSchedulerTracer()`) in its `onStartup()` method, then anything logged through that object is also written to the `i2_Scheduler.log` file.

Enabling sharing to groups of users

By default, i2 Analyze is configured so that users with the `i2:Share:Artifacts` permission can share artifacts such as saved Visual Queries with individual users, but not with groups. Any group that you want to be a target for sharing must be configured appropriately in the i2 Analyze Server Admin Console (where you can also create custom user groups), or through provisioning.

Before you begin

To control which groups are available as targets for sharing, you must have access to the **User group management** app in the admin console. In other words, you must be a member of a system group that has the `i2:Administrator` or the `i2:Administrator:Groups` permission. For more information, see Enabling access to the admin console.

If you're the first visitor to the **User group management** app, you see only the system groups that the user registry defines. By default, these groups are not available as targets for sharing. On subsequent visits, you and other users can also see any custom groups that have been created, as well as the sharing settings for all groups.

Users of i2 Notebook and Analyst's Notebook can share artifacts with the members of a group only if the group is **Available for sharing**, *and* it's visible to them. The latter is always true if the group is **Visible to all users**, but if it's not, the user must be a member of the group (or a sharing administrator) to see it.

About this task

This task describes how to create and configure custom user groups in the i2 Analyze Server Admin Console, and how to enable sharing to system and custom groups.

Procedure

To examine the user groups in your deployment of i2 Analyze, and to create and configure custom groups:

1. Open a web browser and navigate to `<i2-Analyze-URL>/admin`, where `<i2-Analyze-URL>` is the URL used to access your i2 Analyze deployment.
2. Log in with your credentials. Then, in the toolbar on the left, click **User group management**.

   The console displays a list of all the user groups in this deployment of i2 Analyze, including an indicator of whether each is a system group or a custom group.

You can select any group and then click **Members** to see the users and groups who are members of that group. For a custom group, you can add and remove members, as described below.

3. To create a custom user group:
   a. Click **New group**.

      The console displays a dialog where you can enter a name and a description for the group, and say whether it should be **Available for sharing** and **Visible to all users**.
   b. Use the list in the center of the dialog to add users and other groups to the new group. You can search for users and groups by name, and you can add multiple users and groups at once.
   c. Click **Create** to close the dialog and create the group. The new group appears in the list of user groups.
4. To change the name or description of a custom user group, select it in the list and click **Edit details**. Make your changes in the dialog that appears, and then click **Save**.
5. To change the members of a custom user group:
   a. Click either the **Add members** or the **Remove members** button in the Members dialog. (The buttons are not present in the dialog for a system group.)
   b. Use the same approach as when creating a group to add and remove users and groups. The changes you make don't take effect until you click **Add** (or **Remove**) to close the dialog.

To change whether users can share artifacts with the members of *any* user group:

1. Select the group in the list on the front page of the **User group management** app. The pane on the right displays information about the group, including its sharing and visibility settings.
2. Use the switches to change the sharing and visibility settings for the group. The changes take effect immediately.

What to do next

When you've created the custom user groups that you need in your deployment, and you've configured the sharing settings for all user groups, you can test the resulting behavior.

Follow the instructions in the Analyst's Notebook documentation to save and share a Visual Query with a group of users, and then log in as one of those users to see if they can see the Visual Query.

If you can't share the Visual Query with the group you expect, check the sharing settings again.

Upgrading i2 Analyze

Different deployments of i2 Analyze can contain different combinations of its component parts. To upgrade a particular deployment of i2 Analyze, you must upgrade the deployment toolkit and then configure and upgrade the deployment itself.

This information relates to the latest release of i2 Analyze.

Troubleshooting and support

- i2 Support

Upgrading to i2 Analyze 4.4.4

To upgrade a deployment of i2 Analyze, you must first upgrade the deployment toolkit that you are using. You then use this upgraded toolkit to upgrade the deployment.

The version of your current deployment determines exactly which path to follow:

- If you are upgrading from i2 Analyze 4.3.5 or earlier, you must first upgrade your deployment to a supported version before upgrading to this one. The earliest version from which you can upgrade directly is 4.4.0.
- If you are upgrading from i2 Analyze version 4.4.0 or later, complete the instructions below.

For more information about changes to supported versions of i2 Analyze, see Configuration and database changes.

Before you begin

Before you upgrade your production deployment, use a pre-production or test environment to verify that you can complete the upgrade process successfully and that you are familiar with the procedure. After you test the upgrade process for your deployment, complete the upgrade in your production environment. If you do not have a pre-production or test environment, you can create one. For more information, see Creating a production deployment.

- If you are upgrading a deployment of i2 Analyze at version 4.4.2 or earlier, the Solr indexing service is upgraded to version 9. After you upgrade to Solr 9, Apache recommends that you reindex your data. While the data is being reindexed the system is offline. In systems where there is a lot of data, this can take some time. During the upgrade process, you are directed to run the toolkit command that reindexes your data.
- Ensure that you back up your deployment before you complete an upgrade. For more information about backing up your deployment, see Backing up a deployment.

About this task

Depending on the scale and complexity of your data, changes of this nature can take time. You might want to plan your upgrade to take place in a period where there is usually no activity.

As part of the i2 Analyze upgrade process, Open Liberty, Solr, ZooKeeper, and Java are updated. You do not need to download and update these prerequisites before you upgrade an existing deployment.

Procedure

To upgrade the deployment toolkit to version 4.4.4:

1. Stop i2 Analyze:
   - In a single-server topology, run `setup -t stop`.
   - In a multiple-server topology, see Stopping and starting i2 Analyze.
2. On each server where the deployment toolkit is installed, make a backup of the i2 Analyze directory, including the toolkit. For example, `i2/i2analyze`.
3. Install the i2 Analyze deployment toolkit.
   a. On each server where the toolkit is installed, remove the `toolkit`, `license`, and `swidtag` directories from the existing installation.

   b. Install the i2 Analyze version 4.4.4 deployment toolkit over your existing i2 Analyze directory. For more information, see Installing i2 Analyze.

4. For each deployment toolkit in your deployment, copy the `configuration` directory that you backed up in Step 2 to the `i2analyze/toolkit` directory of the upgraded deployment toolkit that you installed in Step 3.
5. If it is not already present, create and populate the `credentials.properties` file. This file must be stored in the location `toolkit/configuration/environment` in each deployment toolkit.

   For more information about the file, see The `credentials.properties` file.

After you upgrade the deployment toolkit, you can use it to upgrade the deployment to version 4.4.4:

1. Upgrade and start i2 Analyze according to the instructions in Upgrading an i2 Analyze deployment.
2. If you are using the IBM HTTP Server, restart it.

Important: If you're upgrading from version 4.4.3 or earlier, the upgrade process includes some automatic changes to the configuration of the deployment, to support user provisioning. The changes have two consequences:

- If you keep a backup of your configuration, take a new backup when the upgrade is complete.
- If you want to use the configuration from an i2 Analyze deployment that was *upgraded* to 4.4.4 in a *new* deployment of i2 Analyze 4.4.4, you must edit it first. Specifically, you need to delete (or set to `false`) the two "`Compatibility`" settings in the `ApolloServerSettingsMandatory.properties` file.

What to do next

If your deployment includes the ETL toolkit, you must upgrade the ETL toolkit to version 4.4.4 after you upgrade the rest of the deployment. For more information, see Upgrading the ETL toolkit.

After you upgrade, you might need to update the configuration of your deployment for any new or modified configuration settings. For more information about new and modified configuration settings, see Configuration and database changes.

When you start the server after you upgrade, extra processing of the data in the Information Store is completed after the upgrade. During this processing, you might not be able to ingest, update, and delete data in the Information Store. For more information, see Information Store processing after you upgrade i2 Analyze.

Upgrading the ETL toolkit

If your deployment of i2 Analyze uses the ETL toolkit, you must upgrade the ETL toolkit to version 4.4.3 separately from the rest of the deployment. To upgrade the ETL toolkit, you must remove the existing version and replace it with the one that is deployed with version 4.4.3 of the i2® Analyze toolkit.

Before you begin

You must upgrade your deployment to version 4.4.3 before you can upgrade your ETL toolkit.

Procedure

1. If you modified the connection properties of your ETL toolkit to connect to a remote instance of Db2®, make a backup of the properties file. Navigate to the `etltoolkit\classes` directory of your

ETL toolkit, and copy the `Connection.properties` file to a location outside of the `etltoolkit` directory. You can now remove the previous ETL toolkit.

2. After you upgrade your deployment, deploy the ETL toolkit from the upgraded i2 Analyze toolkit at version 4.4.3. For more information about deploying the ETL toolkit, see Deploying the ETL toolkit.
3. Update the `Connection.properties` file in the new ETL toolkit with the `db.installation.dir` property and value from the backup `Connection.properties` file from your previous ETL toolkit.

Results

The ETL toolkit is upgraded to version 4.4.3 and ready for use by your ETL logic to modify the Information Store.

Upgrading i2 Analyst's Notebook

To take advantage of some of the latest features of i2 Analyze, users must have the latest version of Analyst's Notebook. To upgrade i2 Analyst's Notebook, you use the Analyst's Notebook Installer.

Procedure

1. Extract the product files from your downloaded distribution of Analyst's Notebook.
2. Using Windows™ Explorer, browse to the root of the distribution and run `setup.exe`.
3. Follow the prompts to complete the upgrade.

   Note: If your previous deployment of i2 Analyze contained an Analysis Repository, the connection to that repository is no longer available after the upgrade. If you had a Local Analysis Repository, the database is retained to allow you to export the data.

Upgrade resources

Depending on the configuration of your deployment, you might need to complete extra tasks to upgrade your system. The extra tasks might need to be completed before or after you upgrade the system.

Upgrading an i2 Analyze deployment

You can use an upgraded i2 Analyze deployment toolkit to upgrade an existing deployment of i2 Analyze. The new features in later versions are available only after the deployment is upgraded successfully.

Before you begin

Complete the instructions in Upgrading i2 Analyze to install the latest version of the deployment toolkit and prepare the environment before you upgrade the deployment.

Note: If your deployment of i2 Analyze uses multiple servers, follow the instructions in Upgrading an i2 Analyze deployment on multiple servers, rather than the procedure in this topic.

Procedure

1. Open a command prompt on the server, and navigate to the `toolkit\scripts` directory of the i2 Analyze toolkit.
2. To upgrade the deployment, run the following command:

```
setup -t upgrade
```

If the `create-databases` attribute in the `topology.xml` file is set to `false`, you must run the scripts to update the database manually.

The scripts are created in the `toolkit\scripts\database\db2\InfoStore\generated\upgrade` directory. Run the scripts in ascending alphanumeric order. The user that you run the scripts as must have permission to drop and create database tables and views.

3. To recreate the Solr collections, run the following command:

   If you run this command, when you start i2 Analyze all of your data in the database will be reindexed. While the data is being reindexed the system is offline. In systems where there is a lot of data, this can take some time. To reschedule the downtime, skip this step for now, and do not run the following command. You can reindex your data at another time by following the instructions in Reindexing data after a major version Solr upgrade.

   ```
   setup -t upgradeAllSolrIndexes
   ```

   When you run this task, you are warned about removing data from the existing indexes. You can continue by pressing **Y** and **Enter**. If a standby index has been created for a collection type, it will be used to reindex the data without modifying the live index.

4. To complete this part of the upgrade and start the application, run the following command:

   ```
   setup -t start
   ```

5. If you are using a deployment with an Information Store and have configured system match rules, run the following command to instruct the system to use the upgraded match index:

   ```
   setup -t switchStandbyMatchIndexToLive
   ```

What to do next

After you upgrade and restart i2 Analyze, return to perform the rest of the instructions to finish upgrading the system.

Upgrading an i2 Analyze deployment on multiple servers

To upgrade i2 Analyze in a multiple-server deployment topology, you must run the commands to upgrade and start the components of i2 Analyze on each server.

Before you begin

Complete the instructions in Upgrading i2 Analyze to install the latest version of the deployment toolkit on the Liberty, Solr, and ZooKeeper servers, and prepare the environment before you upgrade the deployment.

About this task

To upgrade i2 Analyze in a multiple server deployment topology, you must provide an upgraded configuration to each deployment toolkit. Then, you can run the commands to upgrade the components of i2 Analyze on each server.

As you follow the procedure, it is important to keep track of which server each command runs on, and whether you need to specify the hostname of that server to the command.

Run all toolkit commands from the `toolkit/scripts` directory in the deployment toolkit on the specified server in your environment.

Procedure

1. Upgrade and copy the i2 Analyze configuration.
   a. Upgrade the i2 Analyze configuration:

      ```
      setup -t upgradeConfiguration
      ```

   b. Provided that all servers have the same configuration, copy the upgraded `toolkit\configuration` to the `toolkit` directory on each Solr and ZooKeeper server in your environment. Accept any file overwrites.
2. Upgrade the ZooKeeper and Solr components of i2 Analyze.
   a. On each ZooKeeper server, run the following command:

      ```
      setup -t upgradeZookeeper --hostname zookeeper.hostname
      ```

      Where *`zookeeper.hostname`* is the hostname of the ZooKeeper server where you are running the command, and matches the value for the `host-name` attribute of a `<zkhost>` element in the `topology.xml` file.
   b. On each Solr server, run the following command:

      ```
      setup -t upgradeSolr --hostname solr.hostname
      ```

      Where *`solr.hostname`* is the hostname of the Solr server where you are running the command, and matches the value for the `host-name` attribute of a `<solr-node>` element in the `topology.xml` file.
3. Upgrade the Information Store database.

   On the Liberty server, run the following command:

   ```
   setup -t upgradeDatabases
   ```

   If the `create-databases` attribute in the `topology.xml` file is set to `false`, you must run the scripts to update the database manually.

   The scripts are created in the `toolkit\scripts\database\db2\InfoStore\generated\upgrade` directory. Run the scripts in ascending alphanumeric order. The user that you run the scripts as must have permission to drop and create database tables and views.
4. Start the ZooKeeper component of i2 Analyze.

   On each ZooKeeper server, start the ZooKeeper hosts:

   ```
   setup -t startZkHosts --hostname zookeeper.hostname
   ```
5. Upload the Solr configuration to ZooKeeper.

   On the Liberty server, run:

   ```
   setup -t createAndUploadSolrConfig --hostname liberty.hostname
   ```
6. Start the Solr component of i2 Analyze.

   On each Solr server, start Solr:

   ```
   setup -t startSolrNodes --hostname solr.hostname
   ```
7. Upgrade Liberty.

   On the Liberty server, run the following command:

   ```
   setup -t upgradeLiberty
   ```
8. Upgrade the Solr collections and indexes.

a. To upgrade the Solr Collections, run the following command on the Liberty server:

```
setup -t upgradeSolrCollections --hostname liberty.hostname
```

b. To upgrade the Solr indexes, run the following command on the Liberty server.

If you run this command, when you start i2 Analyze all of your data in the database will be reindexed. While the data is being reindexed the system is offline. In systems where there is a lot of data, this can take some time. To reschedule the downtime, skip this step for now, and do not run the following command. You can reindex your data at another time by following the instructions in Reindexing data after a major version Solr upgrade.

```
setup -t upgradeAllSolrIndexes --hostname liberty.hostname
```

When you run this task, you are warned about removing data from the existing match index. You can continue by pressing **Y** and **Enter**. The data is indexed in any standby indexes that has been created.

9. Start Liberty.

On the Liberty server, run the following command:

```
setup -t startLiberty
```

10. If you are using a deployment with an Information Store and have configured system match rules, run the following command to instruct the system to use the upgraded match index:

```
setup -t switchStandbyMatchIndexToLive
```

What to do next

After you upgrade and start i2 Analyze, return to perform the rest of the instructions to finish upgrading the system.

Configuration and database changes

As a result of an upgrade, you might need to consider configuration or database changes that are new or different compared with previous versions of the software. Depending on the default behavior, you might want to modify the configuration to meet your requirements after you upgrade the deployment.

Version 4.4.4

The following changes are introduced at i2 Analyze version 4.4.4:

Java

The minimum supported version of Java for Liberty, Solr, and ZooKeeper is now 17.0.11. Any extensions that you create and deploy for i2 Analyze 4.4.4 must be compatible with this Java version.

Important: The RC4-HMAC Kerberos encryption type was deprecated at Java 11.0.17. If your deployment uses SPNEGO with Kerberos authentication, you must update your Kerberos configuration. See the Java release notes for more information.

User and group management

i2 Analyze 4.4.4 introduces new features for managing users and groups. Administrators have more control over the users and groups that are provisioned from user registries and identity providers, and they can also create and edit custom user groups through the admin console. See User provisioning for more details.

Save and share feature

Users can save and share artifacts such as visual query definitions with other users and groups. See Saving and sharing for more information.

i2 Notebook importer

This is the initial release of the importer for charts in the i2 Notebook web client.

Minimum supported version for upgrade

The minimum version of i2 Analyze that you can upgrade from is now 4.4.0.

Supported component versions

- Postgres version 16 is now supported.
- The minimum supported version for Liberty is now 24.0.0.6.
- The minimum supported version for Solr is now 9.6.1.
- The minimum supported version for ZooKeeper is now 3.9.2.

New command access control permissions

Permission for users to share saved artifacts through the new sharing feature is controlled by a new permission:

`i2:Share:Artifacts`

Users with this permission can share artifacts with other users. Users without this permission can still see artifacts that have been shared with them, and save artifacts for themselves.

The existing `i2:Administrator` permission has been extended with new sub-permissions to provide more fine-grained access control, and to control access to new functionality:

`i2:Administrator:Indexing`

Controls access to the REST API endpoints for viewing the status of the Information Store index, and for clearing and rebuilding it if necessary.

`i2:Administrator:Connectors`

Controls access to the REST API endpoints for managing connectors, and to the gateway status and type conversion apps in the admin console. Users with this permission are automatically granted the `i2:Connectors` permission.

`i2:Administrator:Groups`

Controls access to the user group management app in the admin console.

`i2:Administrator:SavedArtifacts`

Users with this permission are granted full access to all the saved artifacts in the system.

Database table changes

- Added new tables to support importing to web charts in i2 Notebook.
- Added new tables to support user and group management in i2 Analyze.
- Added new tables to support sharing saved artifacts in i2 Analyze.
- Removed the `IS(_Core).User_Principals` table, which is no longer used for user management.

Behavior changes

- The "sensitive" headers provided by the i2 Connect SPI now include two new headers: one that contains the user's ID, and (when available) one that contains the user's OIDC access token.
- The application now supports a timeout for long-running Visual Queries. By default, the timeout is set to 240 minutes (four hours). See Controlling Visual Query timeout for details on configuring the timeout.
- When validating schema files, duplicate display names in the property types of an item type now produce a warning, because Analyst's Notebook uses the display name to identify the property type.

Version 4.4.3

The following changes are introduced at i2 Analyze version 4.4.3:

Java

The minimum supported version of Java for Liberty, Solr, and ZooKeeper is now 17.0.9. Any extensions that you create and deploy for i2 Analyze 4.4.3 must be compatible with this Java version.

Important: The RC4-HMAC Kerberos encryption type was deprecated at Java 11.0.17. If your deployment uses SPNEGO with Kerberos authentication, you must update your Kerberos configuration. See the Java release notes for more information.

Apache Solr

The minimum supported version of Apache Solr is now 9.4.0. This is a major version change from Solr 8, which was included with previous releases.

Important: In Windows deployments, the Solr password can no longer contain any of the following special characters: <, >, |, &, ^, !, %, ". If your deployment uses any of these characters, you must change the password *before* you upgrade. For more information, see Specifying the deployment credentials.

Among other changes, the Solr autoscaling framework has been removed, and shard placement is now configured through replica placement plugins. For further details, see Reindexing data after a major version Solr upgrade.

Apache ZooKeeper

The minimum supported version of ZooKeeper is now 3.8.3. Among other changes, ZooKeeper now uses the Logback logging framework. Any Log4j configuration for ZooKeeper is ignored.

Open Liberty

The minimum supported version of Open Liberty is now 23.0.0.9.

Improvements to the indexing pipeline

For i2 Analyze deployments that include the Information Store, the robustness of the indexing pipeline has been improved.

If a batch of data cannot be indexed because of invalid data, that batch is skipped and added to a separate queue so that indexing can continue. If indexing fails, you can find the details of faulty data in the Information Store view named `IS_Public.Indexing_Failed_Batches`. Note that failed batches can cause inconsistent result counts until you repair and reindex the data.

The REST API also introduces endpoints for clearing the search index and for dealing with failed indexing batches. See the documentation for the `clean` and `retryfailed` endpoints for more information.

Database changes

The Information Store's upgrade views now provide more information about the progress of ongoing upgrades. See Information Store processing after you upgrade i2 Analyze for more information.

Deployment display names

The application can now be configured with an optional display name that will be displayed in Analyst's Notebook Premium and the i2 Notebook web client, making it easier for users and administrators to distinguish between different deployments.

The display name is configured through the `DeploymentDisplayName` setting in `DiscoServerSettingsCommon.properties`.

Custom types in Analyst's Notebook

The latest version of Analyst's Notebook Premium supports the creation of custom i2 Analyze item and property types for charts. In order for users to benefit from this functionality, you must grant them the `i2:CustomTypes` command access control permission. See Configuring command access control for more information.

Security schema validation

Empty `<Permissions>` elements in the security schema now result in validation errors.

Version 4.4.2

The following changes are introduced at i2 Analyze version 4.4.2:

Java

The minimum supported version of Java for Liberty, Solr, and ZooKeeper is now 11.0.19. Any extensions that you create and deploy for i2 Analyze 4.4.2 must be compatible with this Java version.

Important: The RC4-HMAC Kerberos encryption type was deprecated at Java 11.0.17. If your deployment uses SPNEGO with Kerberos authentication, you must update your Kerberos configuration. See the Java release notes for more information.

Minimum supported version for upgrade

The minimum version of i2 Analyze that you can upgrade from is now 4.3.3.0.

PostgreSQL `Collation_Locale`

If your deployment uses PostgreSQL, and you are upgrading from version 4.4.1.0, be aware that the default `Collation_Locale` setting was incorrect. This issue can only be resolved by dropping the database, changing the setting, re-creating the database, and re-ingesting the data.

If your deployment was initially created with the 4.4.1.1 fix pack, or you have already applied the fix pack and re-created your database, then you do not need to take any action.

Information Store schema validation

In i2 Analyze deployments that include the Information Store, the server performs additional validation on deployment and upgrade of the schema. Specifically, it checks for the case where two property types in the same item type have been assigned the same *semantic* property type.

At this version, this invalid state generates a warning in the output from the `deploy`, `upgrade`, and `updateSchema` toolkit tasks. In the future, the warning will become an error. For information about fixing the problem, see Resolving deployment errors.

Note: Previous versions of i2 Analyze included example schemas that suffered from the problem described above. For deployments that use those schemas - or schemas that were originally based upon them - the server repairs the schema dynamically, at runtime.

Information Store security schema changes

i2 Analyze supports a greater range of changes to the security schema *after* deployment, and there are new extension points that allow dynamic modification of the security schema.

NONE access level warning

In a security schema, specifying the `NONE` access level in a permission has no effect, because "no access" is the default state.

At this version, specifying `NONE` is invalid. Using it generates a warning in the output from the `deploy`, `upgrade`, and `updateSchema` toolkit tasks, and from the server. In the future, the warning will become an error. For information about fixing the problem, see Resolving deployment errors.

Removal of dimension values allowed

Dimension values can now be removed from security dimensions without requiring data to be re-ingested or re-indexed. For more details of permitted changes, see Configuring the security schema.

Extension points added

SPI extension points have been added that enable dimension values and permission assignments to be configured in code, as well as in the security schema XML file. For more details and an example of how to implement these SPIs in Java, see the dynamic security example in i2 Analyze Developer Essentials.

Support for Microsoft SQL Server 2022 and recent operating systems

i2 Analyze now supports Information Store deployments on Microsoft SQL Server 2022 as well as 2019 and 2017.

Note: For Windows installations only, the default data directory is version-specific. For SQL Server 2022, it is `C:/Program Files/Microsoft SQL Server/MSSQL16.MSSQLSERVER/MSSQL/DATA`; earlier versions use `MSSQL15` and `MSSQL14` in the same path. In a new deployment, ensure that the `db.database.location.dir` property in `environment.properties` is set correctly for the version of SQL Server you're targeting.

i2 Analyze also adds support for being deployed on Windows Server 2022 and Red Hat Enterprise Linux (RHEL) v9.

Support for record sharing

This release of i2 Analyze introduces the ability for users to share groups of selected records with their peers. On the server, this feature requires the creation of a new Solr collection named `recordshare_index`, and introduces new Command Access Control permissions.

The deployment toolkit automatically configures the Solr collection with default settings as part of upgrade.

By default, users are not able to share records with each other. Instead, they must be granted permission to do so. For details of how to grant permission, see The command access control file.

Apache Solr

The version of Apache Solr included in the release is still 8.11.2, but it has been patched to update the versions of Apache Commons Text and Apache Commons Configuration to address CVE-2022-42889 and CVE-2022-33980.

Note: The CVEs were not exploitable in Solr, but still caused vulnerability scan warnings. The toolkit identifies the version of Solr as `8.11.2-patch-1` and installs it as part of upgrading. You can also install it manually by running `setup -t installSolr`.

Highlight query categories

This release of i2 Analyze adds optional categories to the definitions of highlight queries. You can use these categories to restrict the visibility of highlight queries to specific groups of users. For more details, see Highlight queries.

No changes are necessary to continue to allow all users to see all highlight queries.

Information Store Visual Query configuration

This release of i2 Analyze allows you to control the contents of the Visual Query palette for Information Store searches. This aligns with palette configuration for external Visual Queries, and enables you to include or exclude specific item types from the palette.

These settings are additive, and no changes are required to existing Visual Query configurations. If you do make changes, you can reload the configuration live, through the admin reload endpoint.

Version 4.4.1

The following changes are introduced at i2 Analyze version 4.4.1:

Open Liberty

The Liberty application server installed by the toolkit is now the Open Liberty distribution, rather than IBM® WebSphere® Liberty.

IBM HTTP Server

Toolkit support for configuring the IBM HTTP Server is now deprecated, and might be removed in a future release. If the `topology.xml` file in your deployment sets `http-server-host="true"`, then then toolkit displays a warning message.

Using an HTTP server or proxy for i2 Analyze is optional. If you do want to use one, you can use any modern HTTP server.

Version 4.4.0

No changes are introduced at i2 Analyze version 4.4.0 that affect upgrade.

Version 4.3.5

The following changes are introduced at i2 Analyze version 4.3.5:

Minimum supported version for upgrade

The minimum version of i2 Analyze that you can upgrade from is now 4.3.1.1.

Improved multiple-line string Visual Query performance on IBM Db2

If your deployment includes an Information Store that runs on IBM Db2, this version includes tooling to improve Visual Query performance with properties that contain multiple-line strings.

Version 4.3.4

The following changes are introduced at i2 Analyze version 4.3.4:

Minimum supported version for upgrade

The minimum version of i2 Analyze that you can upgrade from is now 4.3.0.0.

Java install location setting moved

The `java.home.dir` setting in the `environment.properties` file is no longer used. The value that was specified for this setting is now used for the `I2A_COMPONENTS_JAVA_HOME` variable in the `setup.in` file.

For more information about how the location of Java is specified, see Specifying the Java distribution.

Additional setting require for client certificate authentication

If your deployment uses client certificate authentication, you must update the `server.xml` in Liberty with a new mandatory attribute. For more information, see Configuring i2 Analyze for client certificate authentication.

Temporary ingestion table space deprecated in Db2

The `IngestionTempTableSpace` setting in the `InfoStoreNamesDb2.properties` file is superseded by the `InternalStagingTableSpace` setting.

Version 4.3.3

The following changes are introduced at i2 Analyze version 4.3.3:

Record matching

If your existing deployment of i2 Analyze included the i2 Connect gateway but not the Information Store, then the upgrade process modifies the contents of `ApolloServerSettingsMandatory.properties` to reclassify your schema as a *gateway schema*. The effect of this change is to modify the identifiers of the item types in the schema.

If your deployment includes match rules files on the server, the upgrade process automatically updates those files to contain the correct item type identifiers. However, if your users have developed local rules files for Find Matching Records, you must edit those files before they reconnect to the server after an upgrade.

On each workstation, follow the instructions to find the `local-fmr-match-rules.xml` file and use the information in Match rules syntax to update the item type identifiers in the file.

Search results filtering

If your existing deployment of i2 Analyze included the i2 Connect gateway but not the Information Store, then the upgrade process modifies the contents of `ApolloServerSettingsMandatory.properties` to reclassify your schema as a *gateway schema*. If your existing deployment also used a results configuration file, then you must update that file to reflect the reclassification.

Follow the instructions in Setting up search results filtering and Understanding the results configuration file to make the necessary changes to the item type identifiers in the file.

Command access control permissions

A new permission can be added to your command access control file. The new permission grants access to the i2 Notebook web client for the Information Store. For more information, see Controlling access to features.

If you do not update your command access control configuration file to include the new permission, or if command access control is not configured in your deployment, access to the feature is denied to all users.

Liberty log location

The Liberty log files from the i2 Analyze application are now located in the `wlp\usr\servers\opal-server\logs\opal-services` directory. Any logs generated from your previous release remain in their current location. Any new log messages are created in the new files in the new location.

For more information about the log files, see Deployment log files.

New Information Store table space in Db2

The `IS_INTSTG_TS` table space is added to the Information Store database. This table space is used to store the internal staging tables that are created as part of the ingestion process. The `InfoStoreNamesDb2.properties` file contains a new `InternalStagingTableSpace` setting.

Version 4.3.2

No changes are introduced at i2 Analyze version 4.3.2 that affect upgrade.

Version 4.3.1 Fix Pack 1

The following changes are introduced at i2 Analyze version 4.3.1.1:

Command access control permissions

New permissions can be added to your command access control file. The new permissions grant access to upload, delete, and read charts with the Information Store. For more information, see Controlling access to features.

If you do not update your command access control configuration file to include the new permissions, access to the features is denied to all users. If command access control is not configured in your deployment, all authenticated users have access to the new features.

Chart storage

For the Information Store database to store charts, the Information Store schema is updated to include the chart item type. After you upgrade the Information Store database, the following tables are added for chart storage:

- `E_ANALYST_S_NOTEBOOK_CH` - the data table, includes the chart properties
- `E_ANALYST_S_NOTEBOOK_CH_BIN` - the binary data table, includes the binary representation of the chart
- `E_ANALYST_S_NOTEBOOK_CH_TXT` - the text data table, includes the text from the chart that can be searched for

Version 4.3.1

The following changes are introduced at i2 Analyze version 4.3.1:

Minimum supported version for upgrade

The minimum version of i2 Analyze that you can upgrade from is now 4.1.7.0.

OpenJDK - Java Development Kit

The JDK that is installed by the i2 Analyze toolkit is now the OpenJDK, instead of the Oracle JDK.

When you upgrade your deployment of i2 Analyze, the Oracle JDK is uninstalled and the OpenJDK is installed.

Deletion view column changes

To incorporate record identifiers, the deletion views are updated to include a new `item_id` column. The content of the existing `record_id` column now contains the record identifier for a record. The `item_id` column contains the item identifier.

If your existing deletion rules use the `record_id` column, you might need to update them to use the `item_id` column. For more information, see Deleting records by rule.

Version 4.3.0

The following changes are introduced at i2 Analyze version 4.3.0:

Database transaction log size increase

The database management system now logs more tables during data movement, which causes an increase to the size of the transaction log. This is more noticeable in deployments that contain large amounts of data.

`InfoStoreNames.properties` file rename

The `InfoStoreNames.properties` file is renamed to `InfoStoreNamesDb2.properties`.

Version 4.2.1

The following changes are introduced at i2 Analyze version 4.2.1:

Configuration fragments

You can deploy the Information Store and i2 Connect in the same deployment of i2 Analyze. To configure a deployment with this topology, some properties files and settings are in a different location:

- If your deployment contained the `opal-services-daod` fragment, it is renamed to `opal-services`. The settings in the `OpalServerSettingsDaodMandatory.properties` file are now in the `DiscoServerSettingsCommon.properties` file.
- If your deployment contained the `opal-services-is` fragment, the `opal-services` fragment is added and the following files are moved to the `opal-services` fragment:
  - `DiscoClientSettings.properties`
  - `DiscoServerSettingsCommon.properties`
  - Results configuration file
  - `visual-query-configuration.xml`

By default, all of the values for the settings in the files are unchanged.

Command access control permissions

New permissions can be added to your command access control file. The new permissions grant access to the export search results to a CSV file and i2 Connect features in Analyst's Notebook Premium. For more information, see Controlling access to features.

If you do not update your command access control configuration file to include the new permissions, access to the features is denied to all users. If command access control is not configured in your deployment, all authenticated users have access to the features except the export to CSV file feature.

Oracle Java Development Kit

The JDK that is installed by the i2 Analyze toolkit is now the Oracle JDK, instead of the IBM® JDK.

When you upgrade your deployment of i2 Analyze, the IBM JDK is uninstalled and the Oracle JDK is installed.

Reindexing data after a major version Solr upgrade

After a major version upgrade to Solr, Apache strongly recommends that the data in Solr is reindexed. If you are upgrading a deployment of i2 Analyze from version 4.4.2 or earlier, the Solr indexing service is upgraded to version 9 from version 8.

While the data is being reindexed the system is offline. In systems where there is a lot of data, this can take some time. During the upgrade process, the instructions direct you to run the toolkit command that reindexes your data. Alternatively you can reindex the data at another time the suits your organization to reduce downtime during the upgrade procedure or when there is less load on the database.

If you do not reindex your data now, you will have to do so if i2 Analyze adopts the next major version of Solr. Until then, i2 will provide support for indexing issues that can be reproduced on Solr 9.

During the upgrade

To reduce the downtime of the system, during the upgrade you can recreate the transient Solr collections. These collections retain data for a short amount of time, which means they can be recreated quickly with no data in them. The transient collections are; highlight, vq, recordshare, and daod.

After you upgrade Solr, run the following toolkit task from the Liberty server to upgrade the transient Solr collections:

```
setup -t upgradeTransientIndexes --hostname liberty.hostname
```

Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

After you have upgraded the transient Solr collections, continue with the rest of the upgrade procedure:

- Upgrading an i2 Analyze deployment on multiple servers.
- Upgrading an i2 Analyze deployment on a single server.

After the upgrade is complete, you can return to upgrade the rest of the collections at a more suitable time.

After the upgrade

After the upgrade is complete, you can recreate and reindex the non-transient Solr collections at times that suit you. The process is more involved to reduce the total downtime of the system.

You do not have to complete the process for all indexes at the same time. Upgrading the match index does not require any downtime.

If your system is deployed on multiple servers, run the `upgrade` tasks from the Liberty server.

Upgrading the chart collection

1. Before you upgrade the collection, stop Liberty by running the following command on the Liberty server:

   ```
   setup -t stopLiberty
   ```
2. To upgrade the chart collection, run the following command on the Liberty server:

   ```
   setup -t upgradeChartIndex --hostname liberty.hostname
   ```

Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.

3. After you upgrade the collection, restart Liberty by running the following command on the Liberty server:

   ```
   setup -t startLiberty
   ```

   When you start i2 Analyze, the data in any upgraded collection is reindexed.

Upgrading the match index

To upgrade the match index, you can build a standby match index and instruct the system to use that index after it has been built.

1. To upgrade the match collection, run the following command on the Liberty server:

   ```
   setup -t upgradeMatchIndex --hostname liberty.hostname
   ```

   Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.
2. After the upgrade of the match index is complete, instruct the system to use the new index by running the following command:

   ```
   setup -t switchStandbyMatchIndexToLive
   ```

Upgrading the main collection

1. Before you upgrade the collection you must stop Liberty. To stop Liberty, run the following command on the Liberty server:

   ```
   setup -t stopLiberty
   ```
2. Before you upgrade the collection, stop Liberty by running the following command on the Liberty server:

   ```
   setup -t upgradeMainIndex --hostname liberty.hostname
   ```

   Where *liberty.hostname* is the hostname of the Liberty server where you are running the command, and matches the value for the `host-name` attribute of the `<application>` element in the `topology.xml` file.
3. After you upgrade the collection, start i2 Analyze:

   ```
   setup -t startLiberty
   ```

   When you start i2 Analyze, the data in any upgraded collection is reindexed.

Information Store processing after you upgrade i2 Analyze

After you upgrade a deployment of some versions of i2 Analyze with an Information Store, extra processing of the data in the Information Store might take place. Analysts can continue to use the system while data is processed in these *online upgrade* operations.

Upgrade tasks

A number of different processes for updating the contents of the Information Store can occur after you upgrade i2 Analyze. Some of these processes are mandatory and automatic, while you can initiate others through toolkit commands.

The tasks that the server might perform depend on the versions of i2 Analyze that you are upgrading from and to. Each task can place additional restrictions on the system until it is complete.

At this version of i2 Analyze, there are no tasks that run automatically when you upgrade from a supported version.

Status and progress reports

The Information Store provides information about the progress of upgrade processes through a set of views that offer different levels of granularity.

Status view

The `IS_Public.Upgrade_Status` view in the Information Store database shows a complete list of tasks that are Complete or Pending (or have Failed):

task_code	description	status	start_time	end_time
VQ_MLS	Converts record details stored in Db2 as CLOBs to VARCHARS to enable visual queries on them	Pending	2019-08-13 15:30:04.49	

The possible values for the **status** column in this and other views are:

- Pending

  The task is either not started, or started but not complete.

  Note: If you stop i2 Analyze while some tasks are Pending, those tasks continue when you next start i2 Analyze.
- Complete

  The task completed successfully.
- Failed

  The task did not complete successfully.

  If a task fails, check the `messages.log` file for messages or stack traces related to online upgrade. Resolve the issues, and then restart the Liberty server.

Progress view

For tasks that are Pending or Failed in the `Upgrade_Status` view, the `IS_Public.Upgrade_Progress` view shows progress by item type:

task_code	description	schema_type	display_name	status	start_time	end_time	latest_log_entry	entries_processed
VQ_MLS	Converts record details stored in Db2...	ET1	Address	Complete	2023-08-13 15:30:05.12	2023-08-13 15:32:48.89		

task_code	description	schema_type_id	display_name	status	start_time	end_time	latest_log_end_time	rows_processed
VQ_MLS	Converts record details stored in Db2...	ET5	Person	Pending	2023-08-13 15:30:05.12		2023-08-13 15:41:12.20	100000
VQ_MLS	Converts record details stored in Db2...	ET6	Document	Pending	2023-08-13 15:30:05.12			

You can use this view to check how many item types processing is complete for, and how many are still to be processed. When a task fails, the view marks the type being processed at the time - and all subsequent types - as Failed.

Even with upgrade tasks broken down on a type-by-type basis, the processing for a type that has a large number of rows can still take a significant amount of time. In such cases, the **latest_log_end_time** and **rows_processed** columns are updated after every 100,000 records. For more information still, you can examine the log view.

Log view

When a task is Pending in the `Upgrade_Progress` view, and it has data in the **latest_log_end_time** and **rows_processed** columns, `IS_Public.Upgrade_Log` provides the following information:

task_code	schema_type_id	display_name	start_time	end_time	duration_seconds	rows_processed	rows_processed_per_se
VQ_MLS	ET5	Person	2023-08-13 15:30:05.12	2023-08-13 15:41:12.20	667.08	100000	149.91

The `Upgrade_Log` view gains a row for each 100,000 processed records, and for the last batch of records that completes processing for a particular type.

You can examine the `Upgrade_Log` view to confirm that the upgrade process is progressing, and at what rate. If you know roughly how many records are in the Information Store, you can also estimate how long the whole process might take.

Note: Rows in the log view for a particular type are removed when processing for that type ends.

Upgrading a customized Information Store

The Information Store is designed to store large amounts of data, and the underlying database can be customized to optimize performance at scale. If you have modified your database in this manner you must also handle the database upgrade separately.

About this task

If you have customized your Information Store, having the deployment toolkit upgrade your database structure automatically is not desirable. However, to upgrade your system, you still need to modify the Information Store to match the newer version.

Note: Depending on the scale and complexity of your data, making changes of this nature can take time. Plan your upgrade to take place in a period of low activity, and back up your system before proceeding.

Procedure

1. Open a command prompt on the server, and navigate to the `toolkit\scripts` directory of the latest i2 Analyze toolkit.
2. To generate the DDL scripts that can you can use to upgrade your Information Store, run the following command:

   ```
   setup -t generateInfoStoreUpgradeScripts
   ```

   The upgrade scripts are placed in `toolkit\scripts\database\<RDBMS>\InfoStore\generated\upgrade`, where `RDBMS` can be `db2`, `sqlserver`, or `postgres`.
3. Evaluate the scripts provided and use them to update your database.

Results

When you have modified your Information Store to match the latest structure, you must ensure that your applications are upgraded without the database upgrade. To do this, ensure that you have set the `create-database` attribute of the Information Store `data-source` to `false` in the `topology.xml` before Upgrading an i2 Analyze deployment.

Download PDFs

i2 Analyze documentation is also available in PDF documents.

- Full i2 Analyze documentation