Version 4 Release 2

IBM i2 Analyze
Connector Developer Guide

Note

Before you use this information and the product that it supports, read the information in "Notices" on page 9.

This edition applies to version 4, release 2, modification 0 of IBM® i2® Analyze and to all subsequent releases and modifications until otherwise indicated in new editions. Ensure that you are reading the appropriate document for the version of the product that you are using. To find a specific version of this document, access the Configuring section of the IBM Knowledge Center, and ensure that you select the correct version.

Contents

Connecting to external data sources

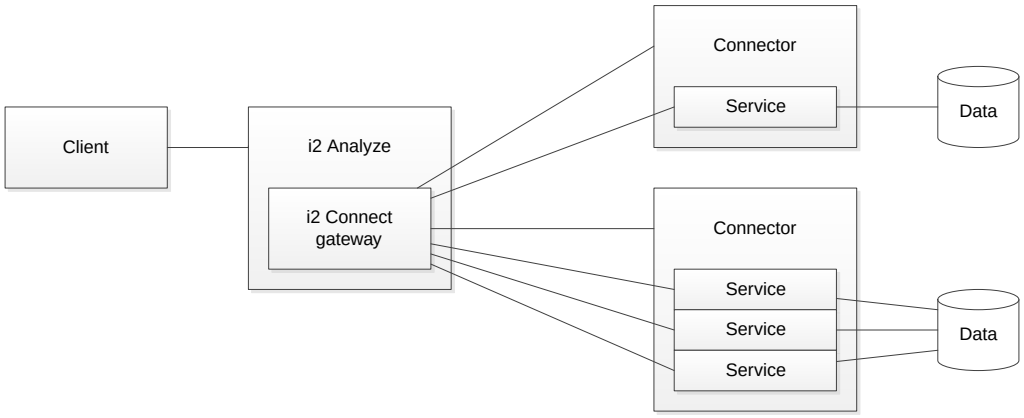
A deployment of IBM i2 Analyze that includes the Opal services can use i2 Connect to query and retrieve data from external data sources. By implementing a connector between i2 Connect and an external data source, you enable i2 Analyze to create and display records that represent the data in that source.

i2 Connect solutions depend on *connectors* that you create to make the bridge between i2 Analyze and external data sources. Conventionally, you write one connector for each source that you want to 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.

IBM i2 Analyze and i2 Connect

At version 4.2.0 of IBM i2 Analyze, you can choose to provide users with access to an Information Store, *or* enable them to query data in one or more external sources. In place of the Information Store, i2 Analyze deployments targeted at external data access include the i2 Connect gateway.

In a complete i2 Connect solution, the gateway provides i2 Analyze with information about what connectors and services are available, and how clients should use them. i2 Analyze passes client requests for data to the gateway, which receives and responds to the requests on behalf of those services.



As well as the code that interacts with an external data source, each connector contains definitions of the services that it supports, and descriptions of how clients should present services to users. The role of the gateway is to retrieve this information from all the connectors that i2 Analyze knows about. It can then help to ensure that calls to and responses from services are formatted correctly.

Put simply, a connector for i2 Connect is an implementation of a REST interface that enables the gateway to perform its role. More specifically, a connector must be able to:

- Respond to requests from the gateway for information about its services
- Validate and run parameterized queries against an external data store
- Convert query results to a shape that the gateway can process into i2 Analyze records

Sometimes, an external data source contains information about entities and links that do not translate to types in the deployed i2 Analyze schema. In that situation, a connector can provide definitions of those types (and how to display them) to i2 Analyze through the gateway.

Writing a connector for i2 Connect requires you to develop an implementation of the REST interface that the gateway uses, and to write queries that retrieve data from an external store. You must also understand the i2 Analyze data model, and be able to convert the retrieved data to the instance of the data model that a particular deployment of i2 Analyze supports.

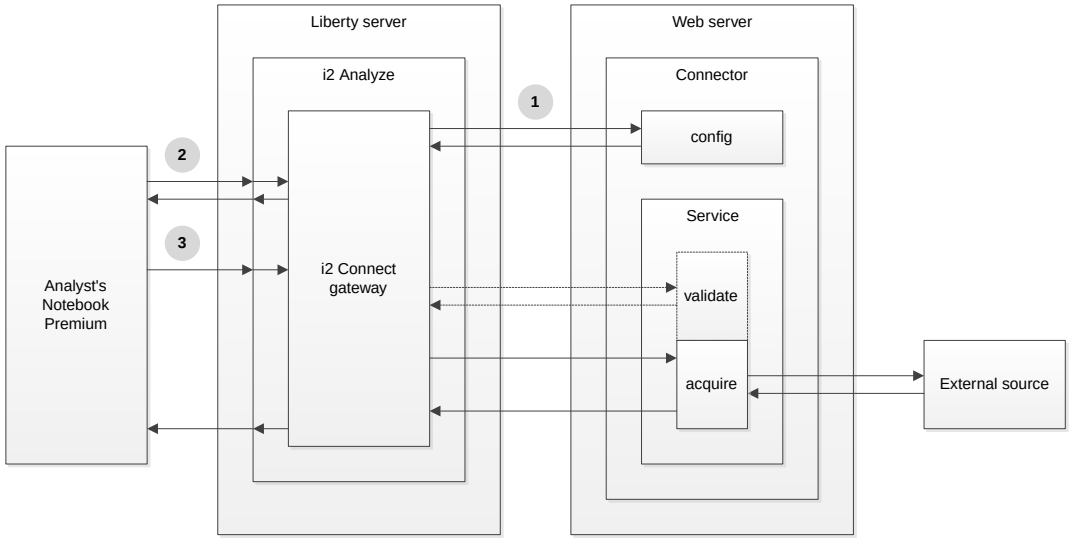
The example that IBM provides in the deployment toolkit demonstrates several of these requirements in a fully functional connector. The remainder of this guide explains how to meet them in more detail.

Architecture of i2 Connect solutions

The simplest possible deployment of IBM i2 Analyze that uses i2 Connect to access an external data source has four parts. The client - Analyst's Notebook Premium - communicates with a deployment of i2 Analyze that includes the i2 Connect gateway. The gateway manages at least one connector, which in turn exchanges data with an external source.

i2 Connect interactions

The following diagram represents the deployment described in the introduction as well as the interactions that take place between them.



The diagram also identifies three of the REST endpoints that connectors for i2 Connect can implement:

Configuration endpoint

The gateway sends a request to the mandatory configuration endpoint to gather information about all the services that the connector supports. All connectors have at least one service, and it is a service that actually implements the validate and acquire endpoints.

Validate endpoint

The validate endpoint is optional, but if the response from the configuration endpoint says that a particular service supports it, then the gateway sends a request to it immediately before a request to the acquire endpoint. The purpose of the validate endpoint is to test whether a request is reasonable before it is executed.

Acquire endpoint

The acquire endpoint is mandatory for all services. The gateway sends a request to it that specifies how the service queries the external source. The service receives data back from the source and places it into the response that it returns to the gateway.

Interaction sequence

The interaction between the gateway and the other parts of an i2 Analyze deployment that includes i2 Connect takes place in three distinct phases, labeled 1, 2, and 3 in the diagram.

1. When i2 Analyze starts up, the i2 Connect gateway sends a request to the configuration endpoint of every connector that is listed in the topology. It caches the information about the connectors and services that it receives in response.
2. When a client connects to i2 Analyze, the latter gets the cached information from the gateway and returns it to the client. The client can then present the queries that the services implement to users, and help users to provide valid parameters to those queries where appropriate.
3. When a user runs a query, i2 Analyze passes it to the gateway for processing. The gateway packages the query into a request to the acquire endpoint on the service in question, preceded by a request to the validate endpoint if the service supports it.

   On receiving the response from the acquire endpoint, the gateway converts the data that it contains into records that i2 Analyze can ultimately return to the client.

Creating a connector for i2 Connect

To create a connector for i2 Connect, you need a data source to query, and a server that supports REST endpoints to query it from. On that foundation, you can build the functionality that takes requests from the i2 Connect gateway, retrieves data from the source, and returns it to the gateway in a shape that is ready for conversion to i2 Analyze records.

About this task

The easiest approach to writing your own connector is to start by getting data from the external source to the i2 Analyze server and then on to the client with as few complications as possible. When you have that mechanism working, you can use it as a base from which to develop the more complex services that you want to implement.

Note: The example connector that IBM provides with i2 Analyze contains two simple services that clients present to users as two separate queries. As you read this documentation, it is helpful also to read the source code for that example.

Writing a parameterless query

All connectors for i2 Connect contain at least one *service* that retrieves information from an external source. By starting with a service that runs a simple query, you can focus on aspects such as the REST endpoint implementations and data structures that are common to all services.

About this task

As you begin to develop a connector for i2 Connect, writing a query that does not require callers to enter parameters makes early testing and development easier. Not only does it simplify the connector, but also it simplifies the act of running the query from the client.

Procedure

There is no fixed set of steps for retrieving data from external data sources, but some of your choices can make subsequent tasks easier:

- If the source allows it, write a query that retrieves a small amount of data.

  At the start of the process, large volumes of data can be distracting.
- If you can, try to make the query retrieve data for entities *and* links.

  Creating the data structures that represent linked entities is an important part of connector development.
- Try to write a query that retrieves data for most of the types that the external source contains.

  No matter how complex your queries eventually become, the code to process data into the right shape for i2 Analyze is unlikely to change.

What to do next

When you implement an acquire endpoint, you can call the query that you developed here directly from that code. Alternatively, during development, you might decide to run the query now, and save the results in a file. Again, this approach simplifies early development, and mirrors the behavior of the example connector that IBM provides.

Extending the i2 Analyze schema

The data that you retrieve from an external source through an i2 Connect service must correspond with entity types and link types that the i2 Analyze schema defines. If there are no types that correspond with the external data, then you must extend the schema as part of connector development.

Before you begin

If you can align the data that your new query retrieves from the external source with existing types in the i2 Analyze schema, then you can skip this task. If you cannot align the data, then you need to add types to the schema.

About this task

i2 Connect permits two approaches to extending an i2 Analyze schema (and its accompanying charting scheme):

1. Add the new types directly to the deployed schema in the usual way. Additive changes to an i2 Analyze schema are always allowed.
2. Create schema and charting scheme *fragments* that you deploy alongside a connector. When you add the connector to your deployment, i2 Connect merges the fragments into the deployment too.

In most cases, updating the deployed schema is the more appropriate approach. Any types that you create are immediately and permanently available throughout the i2 Analyze deployment. When you use IBM i2 Schema Designer to edit the schema, you also gain protection against creating duplicate identifiers.

Creating fragments might be appropriate if you are developing a connector for use in multiple i2 Analyze deployments. In that case, the new types travel with the connector that retrieves data for them. However, you must ensure not only that the type identifiers do not clash with existing identifiers, but also that the types themselves are distinct from existing ones. Having two different types that are both named Person is likely to confuse your users!

Procedure

- If you decide to extend the deployed schema, you can do so in the same way that you would add types to it for any other reason:
  a) Open the XML file that contains the deployed i2 Analyze schema in Schema Designer.
  b) Add entity and link types to the schema that correspond to data from the external store.
  c) Add configuration for the new entity and link types to the charting scheme.
  d) Update the i2 Analyze deployment with the modified schema and charting scheme.

  For more information, see the Schema Designer user guide, and the documentation on modifying the i2 Analyze schema.
- If you decide to create fragments, then you can still use Schema Designer, but after that the process changes:
  a) Open or create an XML schema file in Schema Designer.
     Starting from a schema for one of the target deployments is helpful, but not essential.
  b) Edit the schema and the charting scheme to add types for the data from the external store, and then save and close the files.
  c) Using the `schema.xml` and `chartingSchemes.xml` files from the example as a guide, copy the elements for the new types from the edited files to new XML files.
  d) Arrange for the XML to be available from endpoints on the server that hosts the connector.
     Again, the example includes a simple demonstration of doing this.

  To complete this approach, you must tell the i2 Connect gateway where to find the endpoints that you created. You provide that information in your implementation of the configuration endpoint.

Implementing a configuration endpoint

The configuration endpoint of a connector for i2 Connect must respond to requests with a JSON structure that describes the services in the connector. The more sophisticated the connector, the more information you need to provide.

Before you begin

Navigate to the source code for the example connector in the `connectors\example_connector` directory of your i2 Analyze distribution. In a text editor, open the `app.js` and `config.json` files that together implement the endpoint.

About this task

When the i2 Analyze server starts, the i2 Connect gateway sends a request to the configuration endpoint. The response that your connector sends back tells the gateway what services are available. i2 Analyze caches this information and provides it to clients when they connect.

The simple service that previous tasks describe does not require users to provide parameters, and therefore does not require the client to display user input controls. If you also decide not to create schema fragments, then the structure that you return from the configuration endpoint must contain only two objects:

- The `defaultValues` object is mandatory. Usually, each service in a connector specifies the entity and link types of the data that it can retrieve. However, it is also valid for a service to omit this information, in which case the connector provides defaults. i2 Connect uses the `entityTypeId` and `linkTypeId` from `defaultValues`.

  You can also use the `defaultValues` object to specify a time zone for any value in retrieved data that does not specify its own time zone.
- The `services` array is also mandatory, and must contain information for at least one service. At a minimum, the information must include a unique identifier, a name, and a description for the service. It must also specify the URL of the acquire endpoint, and that this particular service does not require a client UI.

Adding a connector to the topology

When you have a skeleton implementation of a configuration endpoint, you can tell i2 Analyze about the connector that it describes by adding it to the topology file. In that file, the `<connector>` element provides the ability to affect the behavior of the connector and the experience of its users.

Before you begin

Navigate to the `examples\configurations\daod-opal\configuration\environment` directory of the deployment toolkit. In a text editor, open the `topology.xml` file for the example, which references the supplied example connector for i2 Connect.

About this task

Adding a `<connector>` element to the topology file ties together the URL of a server that might host several connectors, and the URL of the configuration endpoint for a particular connector.

Procedure

The topology file that IBM provides to demonstrate i2 Connect includes the `<connectors>` element. Adding a connector to the topology means adding a child to that element.

1. Open the `topology.xml` file for the target deployment in a text editor.
2. Add a `<connector>` element to the `<connectors>` element, according to the following syntax:

```
<connectors>
  <connector id="ConnectorId"
             name="ConnectorName"
             base-url="Protocol://HostName:PortNumber"
             configuration-url="Protocol://HostName:PortNumber/Path"/>
</connectors>
```

ConnectorId must be unique within the topology file, while *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 as the stem for the URLs that you specify in the configuration, while `configuration-url` is the only optional attribute.

By default, the gateway attempts to retrieve the configuration from <*base-url*>/config. You can change this behavior by specifying a different URL as the value of `configuration-url`.

3. Save the file and restart the i2 Analyze server.

   When the server restarts, the i2 Connect gateway makes its request to the configuration endpoint. If that endpoint (or any of the endpoints in the retrieved configuration) is not available, the result is not a fatal error. Rather, the i2 Analyze server logs the problem, and users see messages about unavailable services in the client application.

Implementing an acquire endpoint

In a connector for i2 Connect, an acquire endpoint must perform two tasks in succession. Its roles are to retrieve data from an external source, and to return that data to the i2 Connect gateway in a form that the latter can use to create i2 Analyze records.

Before you begin

Refer again to the `app.js` file for the example connector in your i2 Analyze distribution. It contains an implementation of the acquire endpoint that includes many of the considerations that this topic presents.

About this task

When a user runs a query against an external data source through their client application, the i2 Connect gateway makes a request to the acquire endpoint of the relevant service. The request always has a *payload* that can contain information (often, parameter values) that the user provided to the query.

In general, your implementation of an acquire endpoint uses the contents of the payload to refine the commands that it sends to the external source. If your service specified a client user interface of type `FORM`, then the identifiers of conditions in the form match the identifiers of conditions in the payload.

For a parameterless query, the payload is empty, and you can focus your development effort on making sure that the response from the acquire endpoint meets the requirements of i2 Connect.

Procedure

Note: There are many different kinds of external data source, and it is not possible to follow the same procedure for all of them. The following steps do not apply in all cases. Rather, they describe tasks that you are likely to require.

1. In IBM i2 Analyze Schema Designer, open the schema file that contains the definitions of the i2 Analyze entity, link, and property types that data from the external source translates to.
2. Make a note of the identifiers of the property types of each of the entity and link types in the retrieved data, and make sure that you know which property types are mandatory.
3. Execute the command against the external data source and store the retrieved data in a structure that you can manipulate easily.
4. If the data from the external source contains information about links between entities, and there are no natural identifiers for the links in the data, you must manufacture them.
5. Create the arrays of entity and link data objects that must appear in the response from the acquire endpoint.
6. Populate the arrays according to the descriptions in the SPI documentation, taking care to ensure that you provide values for all mandatory properties.

Starting and testing a connector

Whenever you restart the i2 Analyze server, you force the i2 Connect gateway to refresh the cache of connector configuration information. When a client next asks for a list of the queries that it can use, any changes that you made are reflected in the list.

About this task

A connector that implements only a configuration endpoint and an acquire endpoint is simple, but entirely functional. When you have developed a connector to this point, you can verify that many of the most important mechanisms are functioning correctly.

Procedure

- After you restart the server, make sure that the client displays all the information that you provided about the connector and the service in its list of queries. Different symptoms imply different causes:
  - If the query does not appear in the list at all, then the problem lies with either the implementation of the configuration endpoint, or the `<connector>` element in `topology.xml`.
  - If the query is in the list but marked as unavailable, then the cause of the problem is either the implementation of the acquire endpoint, or the specification of that endpoint in the response from the configuration endpoint.
  - If the displayed information about the query is faulty or incomplete, look again at the definition of the corresponding service in the response from the configuration endpoint.
- Provided that the query that the new connector implements appears correctly in the client, you can run it and view the results that it returns.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Limited Hursley House Hursley Park Winchester, Hants, SO21 2JN UK

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.