Informatica®

Informatica® Cloud Data Integration
July 2024

Components

Informatica Cloud Data Integration Components
July 2024

© Copyright Informatica LLC 2019, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, Informatica Cloud, Informatica Intelligent Cloud Services, PowerCenter, PowerExchange, and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at https://www.informatica.com/trademarks.html. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2024-07-12

Table of Contents

Preface 7

Informatica Resources 7

- Informatica Documentation 7
- Informatica Intelligent Cloud Services web site 7
- Informatica Intelligent Cloud Services Communities 7
- Informatica Intelligent Cloud Services Marketplace 7
- Data Integration connector documentation 8
- Informatica Knowledge Base 8
- Informatica Intelligent Cloud Services Trust Center 8
- Informatica Global Customer Support 8

Chapter 1: Components 9

Chapter 2: API collections 11

Creating an API collection 12

Viewing mapping schemas 12

Synchronizing a REST API request 12

Chapter 3: Business services 13

Defining a business service 13

Chapter 4: File listeners 15

File listeners in file ingestion and replication tasks 15

- File event reliability 16
- File listener job resiliency 17

File listeners in taskflows 17

File listeners in B2B Gateway partner flows 18

Behavioral differences in file listeners 18

Configuring a file listener 19

- Configuring file listener for a server source type 21
- Configuring file listener for a connector source type 22
- Add Parameters 24

Start and stop a file listener manually 27

- Starting and stopping a file listener 28

Chapter 5: Fixed-width file formats 29

Creating a fixed-width file format 29

Chapter 6: Hierarchical mappers 33

Statements 35

Statement types. . . . 35
Statement properties. . . . 37
XPath expression editor. . . . 38
Creating a hierarchical mapper. . . . 39

Chapter 7: Hierarchical schemas. . . . 41

Choosing a sample or schema file. . . . 41
Creating a hierarchical schema. . . . 42

Chapter 8: Industry data service customizer. . . . 43

Message definition. . . . 43
Message structure. . . . 43
Global and positional settings. . . . 44
Customizing an industry-standard message. . . . 44
Adding an element to the message structure. . . . 45
Editing element properties. . . . 45
Adding enumerations. . . . 46
Deleting an element from the message structure. . . . 46
Message properties. . . . 46
HIPAA message properties. . . . 46
HL7 message properties. . . . 48

Chapter 9: Intelligent structure models. . . . 51

Using intelligent structure models in mappings. . . . 52
Using intelligent structure models in mappings in advanced mode. . . . 52
Using intelligent structure models in data engineering mappings. . . . 53
Using intelligent structure models in B2B Gateway inbound partner flows. . . . 53
Intelligent Structure Discovery process. . . . 53
Inputs for intelligent structure models. . . . 54
Output group definition. . . . 56
Simplify Excel. . . . 57
Repeating groups. . . . 57
Primary and foreign keys. . . . 59
Data drifts. . . . 60
Unassigned Data. . . . 60
Maximum record size. . . . 61
Creating an intelligent structure model. . . . 61
Exporting an intelligent structure model. . . . 63
Intelligent structure model example. . . . 63
Use case. . . . 64
Troubleshooting intelligent structure models. . . . 66

Chapter 10: Refining intelligent structure models. 67

Intelligent structure model views. 68
Working with the visual model. 69
Viewing the relational output. 70
Working with relational output. 71
Finding nodes in a structure. 71
Viewing and performing actions on node data. 72
Editing the nodes. 72
Working with repeating groups. 75
Adding document identifiers to data in a model. 75
Adding prefixes and suffixes to field names. 76
Performing actions on multiple nodes. 76
Enriching an existing model with new samples. 77
Edit the structure of Microsoft Excel input. 78
- Transposing a table. 79
- Switching between a table and name-value pairs. 79
- Defining table headers. 80

Testing the output. 80

Chapter 11: Mapplets. 81

Active and passive mapplets. 81
Mapplet input and output. 81
- Mapplet input. 82
- Mapplet output. 82

Transformation names. 82
Parameters in mapplets. 83
Creating a mapplet. 83
Filtering the transformation palette. 84
Editing a mapplet. 85
- Changes that affect dependencies. 85
- Synchronizing a mapplet. 86

Data classifications. 86
- Adding data classifications 86

Validating a mapplet. 87
PowerCenter mapplets. 87
- Active and passive PowerCenter mapplets. 88
- Stored Procedures in mapplets. 88
- PowerCenter XML files for mapplets. 88
- PowerCenter mapplets in Data Integration tasks. 89
- Configuring a PowerCenter mapplet. 90

Chapter 12: Saved queries. 91
Saved query syntax. 91
Using saved queries in synchronization tasks. 92
Using saved queries in SQL transformations. 93
Configuring a saved query. 93
Chapter 13: Shared sequences. 95
Shared sequence properties. 95
Number of reserved values. 96
Creating a shared sequence. 97
Using a shared sequence. 97
Resetting a shared sequence. 97
Chapter 14: User-defined functions. 98
Creating user-defined functions. 98
User-defined function general properties. 99
User-defined function arguments. 99
User-defined function expression. 100
Editing and deleting user-defined functions. 100
Creating expressions with user-defined functions. 100
Parameterizing user-defined functions. 101
Data classifications. 102
Adding data classifications. 103
Index. 104

Preface

Use *Components* to learn about reusable assets that you can create in Data Integration. Learn how to create components and add them to transformations, mappings, and tasks.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at http://www.informatica.com/cloud. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

https://network.informatica.com/community/informatica-network/products/cloud-integration

Developers can learn more and share tips at the Cloud Developer community:

https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

https://marketplace.informatica.com/

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit https://docs.informatica.com.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at https://www.informatica.com/trust-center.html.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The Informatica Intelligent Cloud Services Status page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, on the Informatica Intelligent Cloud Services Status page, click **SUBSCRIBE TO UPDATES**. You can choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Global Support Center through the Informatica Network or by telephone.

To find online support resources on the Informatica Network, click **Contact Support** in the Informatica Intelligent Cloud Services Help menu to go to the **Cloud Support** page. The **Cloud Support** page includes system status information and community discussions. Log in to Informatica Network and click **Need Help** to find additional resources and to contact Informatica Global Customer Support through email.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at https://www.informatica.com/services-and-training/support-services/contact-us.html.

CHAPTER 1

Components

Components are assets that support mappings and tasks.

You can use the following components:

API collections

Create REST API requests to use in the Machine Learning transformation.

For more information about API collections, see Chapter 2, “API collections” on page 11.

Business services

Define a business service to use in Web Service transformations.

For more information about business services, see Chapter 3, “Business services” on page 13.

File listeners

Create a file listener that listens to files on a specific location and notifies subscribers when files arrive at the location and when files in the location are updated or deleted.

For more information about file listeners, see Chapter 4, “File listeners” on page 15.

Fixed-width file formats

Configure reusable formats that you can use for fixed-width flat file sources and targets to enhance readability.

For more information about fixed-width file formats, see Chapter 5, “Fixed-width file formats” on page 29.

Hierarchical mappers

Map a hierarchical schema, such as the schema for an industry-standard message, to another hierarchical schema.

For more information about hierarchical mappers, see Chapter 6, “Hierarchical mappers” on page 33.

Hierarchical schemas

Upload an XML schema, XML sample file, or JSON sample file to use in mappings that include a Hierarchy transformation.

For more information about hierarchical schemas, see Chapter 7, “Hierarchical schemas” on page 41.

Industry data service customizer

Customize an industry-standard message and publish it as a custom data service to the data services repository.

For more information about industry data service customizers, see Chapter 8, “Industry data service customizer” on page 43.

Intelligent structure models

Create or export a model for the following use cases:

- Create a model and use it in a Structure Parser transformation or in a Hierarchy Builder transformation.
- Create a model and use it in advanced mode.
- Export a model and use it in a Data Engineering Integration mapping.
- Create a model and use it to process partner messages in B2B Gateway.

For more information about intelligent structure models, see Chapter 9, “Intelligent structure models” on page 51.

Mapplets

Define reusable transformation logic to use in Mapplet transformations in one of the following ways:

- Create a mapplet in Data Integration .
- Upload a PowerCenter mapplet XML file or an SAP mapplet to extend transformation logic in synchronization tasks and mappings.

For more information about mapplets, see Chapter 11, “Mapplets” on page 81.

Saved queries

Create reusable custom queries that you can use as a source in synchronization tasks.

For more information about saved queries, see Chapter 12, “Saved queries” on page 91.

Shared sequences

Define a reusable sequence to use in multiple Sequence Generator transformations.

For more information about shared sequences, see Chapter 13, “Shared sequences” on page 95.

User-defined functions

Create reusable functions that you can use in transformation expressions and in other user-defined functions.

For more information about user-defined functions, see Chapter 14, “User-defined functions” on page 98.

Create components on the **New Asset** page.

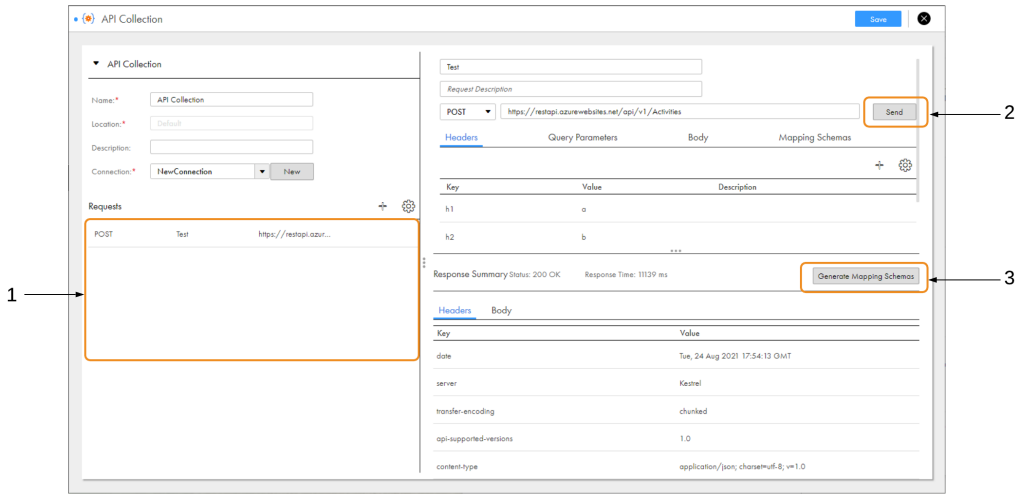
CHAPTER 2

API collections

Create an API collection to store REST API requests to use in the Machine Learning transformation. Requests must use a REST V3 connection.

When you add a new request to the API collection, you enter the request details, test the API, and use the request and response to generate mapping schemas.

The following image shows an API collection:



1. View REST API requests available in the collection.
2. Test the request.
3. Generate mapping schemas based on the request and response.

An API collection provides the following mappings schemas to transformations:

- Request schema. Used to map incoming fields in the transformation to request fields in the REST API.
- Response schema. Used to map response fields from the REST API to output fields in the transformation.

Creating an API collection

Create an API collection and select a default REST V3 connection to test requests. You can use a different connection when you add the API collection to a transformation.

To create a request, add a new request to the API collection and configure the request details. Append path parameters to the endpoint URL. Then, send the request to test it.

After a request in the API collection receives a response, you can generate the mapping schemas. The API collection uses the request source to generate the request schema, and it uses the response source to generate the response schema.

If you use the API collection in a serverless runtime environment and you want to configure TLS to authenticate REST APIs, see the Administrator help for details. To configure TLS for a non-serverless runtime environment, contact Informatica Global Customer Support.

Viewing mapping schemas

View mapping schemas on the **Mapping Schemas** tab. Transformations use the mapping schemas to interact with the REST API.

Mapping schemas ignore empty structures. For example, the address object in the following response is ignored in the response schema:

```
{
"ID":1,
"address":{}
}
```

Note: The response schema can't have a nested child field with the same name as the parent field. If a nested child field shares a name with the parent field, the metadata isn't propagated correctly to the response fields in the Machine Learning transformation.

Synchronizing a REST API request

Synchronize a REST API request to update the request name and the mapping schemas in the Machine Learning transformation.

1. Open the Machine Learning transformation that uses the REST API request.
2. On the **Model** tab, click **Synchronize**.
3. Correct any resulting errors in the transformation logic.

To update request details such as the endpoint URL or header parameters, create a new request in the API collection and use the new request in the Machine Learning transformation.

CHAPTER 3

Business services

A business service is a web service with configured operations. Define a business service to add operations to the Web Services transformation in the Mapping Designer.

Define business services and store them in your project folders. You can use business service definitions in multiple mappings.

Defining a business service

To define a business service, perform the following steps:

1. Click **New** > **Components** > **Business Services** and then click **Create**.
2. Enter a name for the business service and select the location to save it.
3. Select the connection that you want to use or create a new one.
4. Optionally, to refresh the connection metadata every time you run the task, enable dynamic refresh.
5. Select the operation you want to use from the web service.
6. If necessary, configure the operation to specify the choice elements and derived type elements for the request and the response.

   If operation components include choice elements or complexType elements where the abstract attribute is true, then you must choose one or more elements or derived types when you configure the operation mapping.

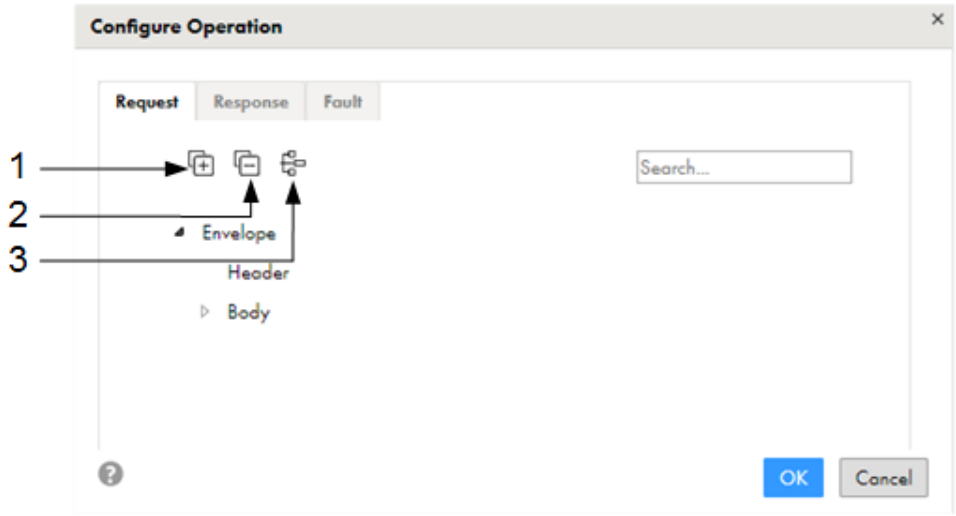
   Optionally, for a complexType element where the abstract attribute is false, you can also select a derived type for a complexType element.

   a. For the operation you want to configure, click **Configure**.

   b. From the **Configure Operation** window, click the **Request**, **Response**, or **Fault** tab and navigate to the node you need to configure.

      Note: If the WSDL uses the anyAttribute element, the element will not appear for the request or the response.

You can click the icons at the top to navigate to the nodes that you need to configure:



1. Expand all nodes.
2. Collapse all nodes.
3. Show only nodes that you must configure. If you select this icon and no nodes expand, then you do not have to choose any choice elements or derived types.

c. Select the choice elements or derived types.

   Ensure that you configure the request and the response for the operation by clicking the relevant tabs.

d. Save the configured operation.

7. Optionally, add more operations and configure the operations if necessary.
8. Save the business service.

   If you have not configured all the required choice elements and derived types, then you cannot save the business service.

CHAPTER 4

File listeners

A file listener listens to files on a defined location. Taskflows, file ingestion and replication tasks, and B2B Gateway partner flows use file listeners to monitor specific folders, and receive notification through a call-back API when a file event occurs. A file event occurs when new files arrive to the monitored folder or the files in the monitored folder are updated or deleted.

You define a file listener that listens to a specific folder and the file pattern. You can define the file events that trigger a notification to the following assets:

- Taskflows
- File ingestion and replication tasks
- B2B Gateway partners

Note: You must have the Read File Listener and Run File Listener privileges to assign a file listener to the assets.

For example, you can define whether file ingestion and replication tasks are notified when new files arrive or files in the monitored location are updated or deleted. You then assign the file listener to a file ingestion and replication task. The file listener listens to source folder, and notifies the file ingestion and replication task when a file arrived, file updated, or file deleted event occurs. You can configure the source as a server event or a connector. The file ingestion and replication task then runs and picks up files from the source folder.

A file listener sends notifications to the user who created it when it starts listening to the folder, when it stops listening to the folder, and if errors occur while it listens to the folder.

The file listener creates a job when the file listener starts and lists the job instance in the file transfer logs page of Monitor. The file listener updates the job when file events occur. The file listener job details appear in the file listener job properties.

A file listener that is not used by a taskflow, file ingestion and replication task, or B2B Gateway partner is not active and does not listen to the defined folder.

File listeners in file ingestion and replication tasks

You can use a file listener as a source and to schedule file monitors in file ingestion and replication tasks.

In file ingestion and replication tasks with the following source types, you can schedule the task to run when it receives notifications from a file listener:

- Local folder
- Advanced FTP V2
- Advanced SFTP V2

- Advanced FTPS V2
- Amazon S3 V2
- Microsoft Azure Data Lake Store Gen2
- Microsoft Azure Data Lake Store V3
- Microsoft Azure Blob Storage V3
- Microsoft Fabric OneLake
- Google Cloud Storage V2
- Hadoop Distributed File Storage (HDFS) V2

Note: For more information on configuring a file ingestion task, see the Data Ingestion and Replication help.

File event reliability

When you use a file listener as a source in a file ingestion and replication task, it creates a file event based on the file listener configuration when new files arrive, when the existing files are updated, or when the files are deleted. The file events are passed to the file ingestion and replication task. This section explains the reliability aspects of handling these file events between a file listener and a file ingestion and replication task.

The file listener handles the events based on the following conditions:

- If the Secure Agent isn't running or there is a temporary network disruption, and file events don't reach the file ingestion and replication task, the file listener queues the events for each file and includes it in the notification of the next file ingestion and replication job. A file ingestion and replication task thus receives a notification about each file at least once. This ensures at-least-once reliability between the file listener and the file ingestion and replication task.

  Note: File events that aren't processed remain in the queue for seven days.
- If multiple events occur, the file listener notifies the file ingestion and replication task with only the last event for each file.
- File events that are in the file listener queue reach the file ingestion and replication task by one of the following methods:
  - When a file ingestion and replication job completes, the Data Ingestion and Replication service makes a pull request to the file listener to check for any queued events. If it finds any events, the service triggers a new ingestion job to process them. The pull request doesn't trigger the processing of files that are already assigned to another concurrent job that runs by the same ingestion task, so only one ingestion job processes a file at any time.
  - If any events aren't picked up by the proactive pull request, for example, if the Secure Agent isn't running when the Data Ingestion and Replication service makes the request, the file listener queues the last event for each file and includes it in the notification of the next file ingestion and replication job.
  - You can also run the file ingestion and replication task manually to pull the failed events.
- When a file event processing fails, the file ingestion and replication task retries to process the failed events. Retry of failed events occurs once automatically and during subsequent file listener notifications.
- The file ingestion and replication task doesn't automatically reprocess file events that are in success or duplicate status.
  You need to manually identify files that aren't successfully transferred to the target due to an error, for example, by using the transfer logs. To resolve the problem, either move the files or modify them manually, so that the file listener picks them up. For example, if the last modified time of a file changes, the file listener identifies the file as updated even if the contents haven't changed.

Example

A file listener is a source in a file ingestion and replication task with 15 file events to transfer to a target. The batch size is five. When the file ingestion and replication task is triggered and complete, the file events are in the following status:

- Five events in the first batch (file 1 to 5): success
- Five events in the second batch (file 6 to 10): failed
- Five events in the third batch (file 11 to 15): unprocessed

The file ingestion and replication task automatically retries to process the five failed and unprocessed events once. When the file ingestion and replication task is complete, the file events are in the following status:

- Five events in the first batch (file 6 to 10): success
- Five events in the second batch (file 11 to 15): failed

The file ingestion and replication task automatically retries to process the five failed events once. When the file ingestion and replication task is complete, the five events in the second batch (file 11 to 15) fails.

You can manually run the file ingestion and replication task to pull the pending five events. If you don't run the file ingestion and replication task manually, the file listener would include the failed events in the notification of the next file ingestion and replication job.

File listener job resiliency

When you run a file listener, a corresponding job is created.

The file listener job handles the Secure Agent availability based on the following conditions:

- If a new version of Application Ingestion and Replication is released, the file listener stops and restarts running on the new version.
- When a Secure Agent is unavailable and is back up within 20 minutes, the file listener resumes running on the same Secure Agent after it restarts.
- If a Secure Agent is unavailable for 20 minutes, the file listener restarts on any running Secure Agents available in the Secure Agent group. This behavior applies if the file listener hasn't reached its end time.
- When a Secure Agent is unavailable for more than 20 minutes in a Secure Agent group, the file listener remains in an unresponsive state. After 200 minutes, its status changes to **Stopped**.

File listeners in taskflows

You can use a file listener in a taskflow when the file listener is configured to listen to connections.

You can use a file listener in a taskflow for the following use cases:

To invoke a taskflow through a file listener

You can invoke a taskflow through a file listener with the connector source type.

Within the taskflow, define the binding type as **Event** and select the file listener as the event source. When you publish the taskflow, the taskflow subscribes to the file listener that is defined in it. When a file event occurs, the file listener invokes the taskflow.

For example, if you configure the file listener to listen for new files on a folder, the file listener invokes the associated taskflow each time a new file arrives in the specified folder.

To orchestrate taskflow execution through file events

You can orchestrate taskflow execution through file events by using the File Watch Task step in a taskflow.

You can add a File Watch Task step to a taskflow to listen to files in a defined location and monitor file events. In the File Watch Task step, you can select an existing file listener with the connector source type. You can use file events to orchestrate taskflow execution.

For example, you can wait for a file to arrive at a particular location and then consume the file in a subsequent step.

File listeners in B2B Gateway partner flows

You can use a file listener to trigger B2B Gateway partner inbound and outbound flows.

When you configure the partner, you select the file listener to trigger the inbound or outbound flow. When you save the partner, the partner subscribes to the file listener. The file listener triggers the flow according to the rules defined in the file listener.

For example, you configure a file listener to listen for new files arriving in a folder and then configure the partner to use the file listener for inbound flows. When the partner puts files into the specified folder, the file listener triggers the inbound flow.

Behavioral differences in file listeners

Certain behavioral differences exist when a file listener is used in file ingestion and replication tasks and B2B Gateway partner flows, and in the File Watch Task step of a taskflow.

The following table describes the behavioral differences in file listeners based on where they are used:

Behavior	File listener used in file ingestion and replication tasks and B2B Gateway partner flows	File listener used in the File Watch Task step of a taskflow
Lifecycle	The file listener runs until the first occurrence of a file event or until the configured end time.	The file listener runs until the first occurrence of a file event or until a timeout occurs. If a file event does not occur, by default, the taskflow waits for 5 minutes or for the overridden value defined in the **Time Out** input field of the File Watch Task step. The maximum time period for which the taskflow waits is 7 days, after which it times out.
End time of run	The latest end time of the run is at 11:55 PM on the configured end date and time zone, and cannot extend to the following day.	The file listener runs until a file event or timeout occurs, and does not depend on any end date or time zone.

Behavior	File listener used in file ingestion and replication tasks and B2B Gateway partner flows	File listener used in the File Watch Task step of a taskflow
Snapshots	- All the file listener instances share the same snapshot. - Snapshots are never deleted. - The file listener run is listed in the **File Transfer Logs** tab in Monitor with the file listener name as the instance name. For example, if the file listener name is **FL_Delete**, the instance name that you need to look for in the **File Transfer Logs** tab in Monitor would be **FL_Delete**.	- Each file listener instance maintains its own snapshot. - Snapshots are deleted immediately after the jobs complete. - Each file listener run is listed in the **File Transfer Logs** tab in Monitor. Append the monitorJobId that you see in the output fields to the file listener name to find the instance name in the **File Transfer Logs** tab in Monitor. For example, if the monitorJobId is **7500** and the name of the file listener is **FL_Arrive**, the instance name that you need to look for in the **File Transfer Logs** tab in Monitor would be **FL_Arrive-7500**.
Start and Stop	You can start and stop a file listener instance from Data Integration or by using the File Transfer REST API resources.	When you run a taskflow that contains a File Watch Task step, the associated file listener instance starts. The file listener instance stops when the first occurrence of a file event or a timeout occurs. When you manually start or stop a file listener instance, the taskflow is not impacted.

Configuring a file listener

Before you create file listeners, verify that the following conditions exist:

- The organization has the appropriate licenses.
- The file listener is running on the secure agent.

Perform the following steps to configure a file listener:

1. To create a file listener, click **New** > **Components** > **File Listener**, and then click **Create**.

   To edit a file listener, on the **Explore** page, navigate to the file listener. In the row that lists the file listener, click **Actions** and select **Edit**.
   When you edit a file listener, such as update the run-time environment, connector type, connection, or listener rule properties, the file listener requires a reset. A warning message appears. Click **Ok** to continue. This action clears the list of previously tracked files.

2. Configure the following file listener details:

Parameter	Description
File Listener Name	Name of the file listener. A file listener name must be unique within the organization. File listener names can contain alphanumeric characters, spaces, and underscores. The names must begin with an alphabetic character or underscore. File listener names are not case sensitive.
Location	Project directory in which the file listener is created.
Description	Optional description of the file listener. Maximum length is 1024 characters.
Status	Status of the file listener: Enabled or Disabled. A disabled file listener does not listen to files on the designated folder.
Runtime Environment	Runtime environment that contains the Secure Agent used to run the file listener.
Source Type	Type of source to which file listener listens. Select one of the following source types: - **Server**. File listener listens to the server. For more information about parameters and file listener rules that you configure for the server, see "Configuring file listener for a server source type" on page 21. - **Connector**. File listener listens to the connection. For more information about parameters and file listener rules that you configure for the connector, see "Configuring file listener for a connector source type" on page 22.

3. Configure the schedule by which the file listener runs:

Parameter	Description
Run	Frequency at which the file listener runs, daily, weekly, or monthly.
Start Date	Date on which the file listener starts running.
End Date	Date until which the file listener runs.
Repeat indefinitely	The file listener runs without an end date.
Start At	Time of day when the file listener starts running.
Check Until	Time of day when the file listener stops running.
Check Every	Frequency at which the file listener checks for files in the folder to which it listens, by seconds, minutes, or hours.
Time Zone	Time zone according to which the file listener runs.

Parameter	Description
Days to Run	Days of the week when the file listener runs when you select to run the file listener weekly.
Day of Month	Day of the month when the file listener runs when you select to run the file listener every month.
Day of Week	Day of the week the file listener runs when you select to run the file listener every month
Week	The week the file listener runs when you select to run the file listener every month. For example, to run the file listener on the fourth Sunday of the month, select **Day of Week**: **Sunday** and **Week**: **Fourth**.

4. Under Failure Management, you can select the following options:
   - **Continue to run on failure**. The file listener continues to retry and run in case of a failure, such as a temporary network disruption.
   - **Send a notification on failure**. Receive a notification if the file listener fails. Enter a list of email addresses to which you want to send notifications if the file listener fails.
     Use commas to separate a list of email addresses.
5. Click **Save**.

Configuring file listener for a server source type

When you select the source type as server in the file listener details, file listener listens to the server events.

You can configure a file listener to listen to events on AS2, HTTPS, and SFTP servers. For example, you can configure the file listener to send notification when the AS2 server receives a file. For more information about configuring file listener details, see "Configuring a file listener" on page 19.

Configure the following parameters and the file listener rules when you select a server source type:

1. Configure the following parameters to define the events that file listener listens:

Parameter	Description
Event Provider	The server type to which the file listener listens. You can configure the file listener to listen to the AS2, HTTPS, or the SFTP server.
Event Type	The type of server event to which file listener listens. The event types that you can define for the AS2 server are: - AS2 Message Receive Failed - AS2 Message Receive Successful The event types that you can define for the HTTPS server are: - HTTPS Upload Failed - HTTPS Upload Successful The event types that you can define for the SFTP server are: - SFTP Upload Failed - SFTP Upload Successful For example, if the event type is SFTP Upload Failed, then file listener sends notifications when the file upload fails on the SFTP server.

2. Configure the following parameters to define listener rules:

Parameter	Description
Key	The event attributes.
Type	The methods to filter the key value. - Contains. Filters keys that contain the value. - Equals. Finds an exact match to run the rule.
Value	Value of the key.

Configuring file listener for a connector source type

When you select the source type as connector in the file listener details, the file listener listens to the connection.

For more information about configuring file listener details, see "Configuring a file listener" on page 19.

1. Configure the following parameters to define the connection:

Parameter	Description
Connection Type	Type of the connection to which the file listener listens.
Connection	Connection to which the file listener listens.

2. Configure the following parameters to define listener rules:

Parameter	Description
Folder Path	Path to the folder to which the file listener listens. **Note:** File listener can access files and directories on network shares with support for NFS and CIFS. You can enter a relative path to the source file system. To enter a relative path, start the path with a period, followed by a slash (./). The path is relative to the source directory specified in the connection.
Pattern Type	Determines the pattern of the file name to which the file listener listens. Select one of the following patterns: - Wildcard - Regex - Indicator File

Parameter	Description
File Pattern	File name pattern to which the file listener listens. - Wildcard. Use wildcard patterns of file name. - Regex. Use a regular expression to match the file name pattern. If you select regex pattern type, consider the following examples: - Use the following syntax to listen to all files except for files whose name contains out, foo, and baz. `^(?!.*(?:out|baz|foo)).*$ all except` - Use the following syntax to listen to all files that have an extension of doc,docx, pdf. `([a-zA-Z0-9\s_\\.\-\(\):])+(.doc|.docx|.pdf)$` - Use the following syntax to listen to all text file except for files whose name contains `out.txt`. `^(?!out).*\.txt$` - Indicator File. Use the file name to which the file listener listens.
Check for files in sub-folders	Indicates whether the file listener checks for files in sub-folders under the folder to which it listens.
Post Action	Determines the action the file listener must perform after the file listener listens to the events. You can select the post action as **Delete** only if the file pattern is an indicator file. Default is None. The following connection types support the Post Action option: - Local folder - Advanced FTP V2 - Advanced FTPS V2 - Advanced SFTP V2 - Amazon S3 V2 - Microsoft Azure Data Lake Store Gen2 - Microsoft Fabric OneLake
Add Parameters	Create an expression to add it as a folder path parameter. For more information, see "Add Parameters" on page 24.
Notify when file	Determines when the file listener must send notifications to the services that are registered to it: - Arrived. Sends notifications when files arrive at the folder to which the file listener listens. - Updated. Sends notifications when files in the folder to which the file listener listens are updated. - Is Deleted. Sends notifications when files in the folder to which the file listener listens are deleted. You can select as many options as required. **Note:** Do not select the **Is Deleted** option if the **Post Action** is **Delete**.
Stop checking if rules are met	The file listener stops listening to the folder when the listener rules are met. For example, if you configure the file listener to send notifications when the files in the folder to which it listens are deleted, the listener stops listening to the folder when the first event of file deletion occurs in the folder. If this option is not selected, the file listener notifies the registered application on events and continues to listen for subsequent events.

Parameter	Description
Check File stability	The file listener verifies that the entire file is copied to the folder to which it listens before notifying the registered services. **Tip:** Select this option if you transfer large files, where the process of writing the files to the folder is not instant.
Stability Check Interval	Time in seconds that a file listener waits to check for file stability. For example, if the stability time is 15 seconds, a file listener verifies the status of the file after every 15 seconds. The stability check interval field is enabled only if you select the **Check file stability** option. The stability check interval ranges between 10 seconds to 300 seconds. Default is 10 seconds.
Notify if files exist on the first run	When the file listener runs for the first time, it sends a notification if files exist in the folder to which it listens. The **Exclude events when the file listener is not running** option is disabled when you select this option.
Exclude events when the file listener is not running	Determines if you want to exclude the file events that occur when a file listener is not running. The **Notify if files exist on the first run** option is disabled when you select this option.

3. Consider the following options to specify a schedule:

1.
   - You can set schedule options to run a file listener daily, weekly, or monthly.
   - You can set schedule options to start a file listener on a specific date and time, and seconds, based on a specific time zone and to run file listener on a reoccurring basis.
4. Under **Failure Management**, select **Continue to run on failure** to run file listener even if the task fails with errors. Select **Send a notification on failure** to receive a notification if the task fails. Enter a list of email addresses to which you want to send notifications if a task fails.

Add Parameters

Using variables you can configure a parameter that a file listener reads from or writes to.

You can use one of the following types of variables to configure a parameter:

- System variables
- User-defined variables

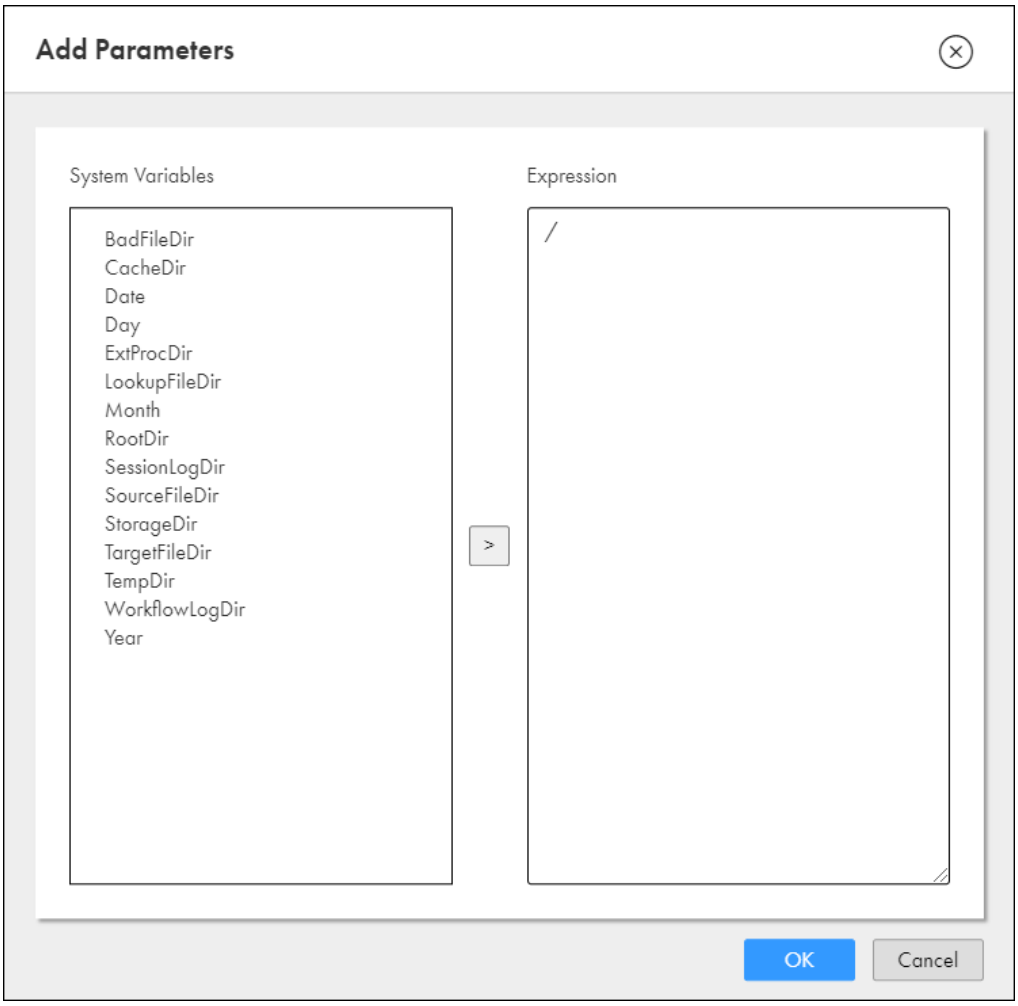
Note: You cannot run a task using the user-defined variable from the user interface. The value of the user-defined variable must be passed using the job resource of the File Ingestion and Replication REST API. For more information, see the REST API help.

Using system variables to add parameters

Use system variables to add parameters to **Folder Path** on the task wizard.

1. Click the icon next to **Post Action** on the file listener task wizard.

   The **Add Parameters** window appears.



2. Select the required variable from the **System Variables** column, and click >. The selected system variable appears on the **Expression** column. Repeat this procedure to select multiple system variables.

   Note: When using a System variable within a task, it should be formatted as `${systemvariablename}`.

The following table describes the system variables:

System Variables	Description	Expression
BadFileDir *	Directory for reject files. It cannot include the following special characters: * ? < > " | ,	${$PMBadFileDir}
CacheFileDir *	The location for the cache file.	${$PMCacheDir}
Date **	The current date in ISO (yyyy-MM-dd) format.	${system.date}
Day **	The day of week	${system.day}
ExtProcDir *	Directory for external procedures. It cannot include the following special characters: * ? < > " | ,	${$PMExtProcDir}
LookupFileDir *	Directory for lookup files. It cannot include the following special characters: * ? < > " | ,	${$PMLookupFileDir}
Month **	Numerical month	${system.month}
RootDir	Root directory accessible by the node. This is the root directory for other service process variables. It cannot include the following special characters: * ? < > " | ,	${$PMRootDir}
SessionLogDir *	Directory for session logs. It cannot include the following special characters: * ? < > " | ,	${$PMSessionLogDir}
StorageDir *	Directory for run-time files. Workflow recovery files save to the $PMStorageDir configured in the PowerCenter Integration Service properties. Session recovery files save to the $PMStorageDir configured in the operating system profile. It cannot include the following special characters: * ? < > " | ,	${$PMStorageDir}

System Variables	Description	Expression
TargetFileDir *	Directory for target files. It cannot include the following special characters: * ? < > " | ,	${$PMTargetFileDir}
TempDir *	Directory for temporary files. It cannot include the following special characters: * ? < > " | ,	${$PMTempDir}
WorkflowLogDir *	The location for the workflow log file.	${$PMWorkflowLogDir}
Year **	Year	${system.year}
* Values are fetched from the Data Integration Server. ** Time zone is the Secure Agent time zone.		

Note: The parameter substitution occurs when you start a task. When a file listener starts and evaluates that the previously watched location is different due to the difference in the parameter value, a delete notification is sent for the files present in the previously watched location. The notifications are not sent if you disable the delete notification option or enable the **Exclude events when the file listener is not running** option.

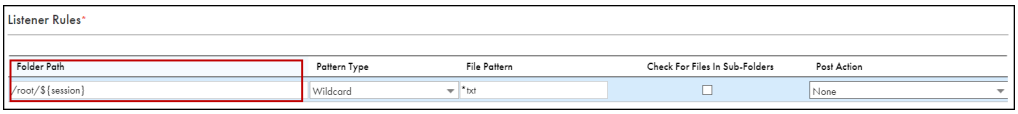
3. Click **OK**.

   The expression appears in the **Folder Path** field.

Using user-defined variables to add parameters

Use user-defined variables to define value for **Folder Path** on the task wizard.

1. Click the **Folder Path** field and enter the variable. The variable should be formatted as `${systemvariablename}`.



2. Click **OK**.

   The expression appears in the **Folder Path**.

Start and stop a file listener manually

A file listener runs at a defined frequency according to the run schedule. If you do not want to run the file listener on a pre defined schedule, you can start or stop a file listener manually.

A file listener listens to files on a defined location from the start time until the end time when you schedule a file listener run.

Consider the following file listener behavior when you start and stop a file listener manually:

- You start a file listener before a scheduled run. The file listener that is triggered manually runs and stops at the start time of the schedule run.
- You start a file listener after the end time of a schedule run. The file listener runs once and stops.
- You cannot start a file listener manually when a file listener is already running according to the run schedule.
- When the agent version is upgraded, the file listener is automatically submitted to the new agent version and the old version is stopped.
- If the agent fails, such as a temporary network disruption, the file listener is automatically submitted to the agent when the agent restarts. Events such as Arrive, Update, and Delete that were submitted when the agent failed are notified by the file listener when the agent restarts.
- You stop an existing file listener run. The file listener stops listening to files for that run. A new run starts at the time of the next schedule time unless the file listener is started manually.
- A file listener fails to start if the file listener is disabled, it is not associated to a task, or if the associated secure agent is down.
- A file listener fails to start if the number of file events such as arrive, update, or delete, exceeds the maximum event limit (10000 files).

Starting and stopping a file listener

You can start and stop a file listener from the **Edit** file listener component page or the **Explore** page using the **Action** menu. Perform the following steps to start or stop a file listener manually.

1. In the **Explore** page, select one or more file listeners and click the **Action** menu. If you want to start all file listener, you can filter the file listener asset in the **Explore** page and click **Select All**.

   The **Start** and **Stop** buttons are enabled.
   If the file listener fails to start, a message with the failure details is displayed.
2. Click **Start**.

   The file listeners validate the configuration and start listening to files defined in the file listener configuration rules.
3. Click **Stop** to stop the file listener. If you want to stop all file listener, you can filter the file listener asset in the **Explore** page and click **Select All**. If the file listener fails to stop, a message with the failure details is displayed.

CHAPTER 5

Fixed-width file formats

You can create and save fixed-width file formats that specify the formatting details for fixed-width flat files.

You can use a fixed-width flat file as a source or target in mappings and mapping tasks. When you select a flat file connection for a Source transformation or Target transformation, you specify the flat file type. If you select the fixed-width flat file type, you select the most appropriate fixed-width file format to use based on the data in the fixed-width flat file.

You can create multiple fixed-width file formats. For example, you might have a fixed-width file format to use for fixed-width flat files that contain quarterly sales data and another fixed-width file format to use for fixed-width flat files that contain inventory data.

If a fixed-width file format does not already exist, you must create a fixed-width file format before you create a mapping or mapping task that uses a fixed-width flat file.

To configure a fixed-width file format, you specify the number of columns and the width, name, and datatype for each column. You can also set advanced fixed-width format properties. For example, you can specify how to handle null characters or specify the default date format for each column.

You can delete fixed-width file formats that you no longer use. You cannot delete a fixed-width file format that is used in a mapping or mapping task.

Creating a fixed-width file format

Create a fixed-width file format so that you can use a fixed-width flat file as a source or target in a mapping.

When you specify format options for a fixed-width file format, use a sample flat file. The data in the sample file appears on the page to help you determine the appropriate format options for your data. The format options that you specify do not alter or save the sample file.

1. Click **New** > **Components** > **Fixed Width File Format** and then click **Create**.

   To edit a fixed-width file format, on the **Explore** page, navigate to the fixed-width file format. In the row that contains the fixed-width file format, click **Actions** and select **Edit**.

2. Enter the following fixed-width file format details:

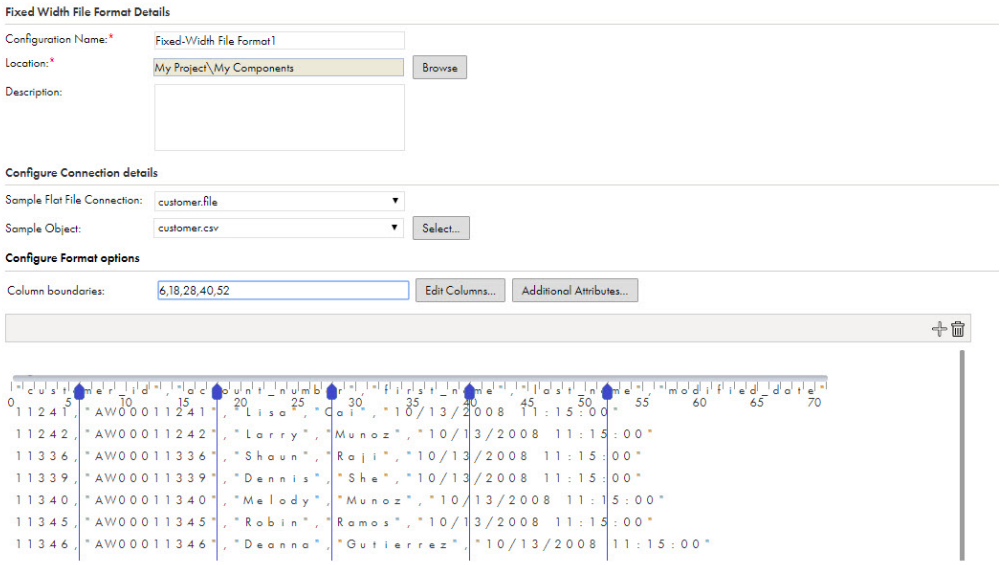
Property	Description
Configuration Name	Name of the fixed-width file format.
Location	Location of the fixed-width file format. Browse to the folder where you want to store the fixed-width file format or use the default location. If the **Explore** page is currently active and a project or folder is selected, the default location for the asset is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.
Description	Description of the fixed-width file format.

3. Specify the following connection details for the sample fiat file:

Property	Description
Sample Flat File Connection	Connection to the sample file that you use to specify the fixed-width format.
Sample Object	Sample file that you use to specify the format options.

4. Configure the column boundaries to fit the data in your sample file. You can enter the boundaries in the **Column boundaries** field or you can position the boundaries using the ruler.
   - To configure column boundaries in the **Column boundaries** field, enter the number of bytes to determine the width of the first column. Then enter boundaries for the subsequent columns. Separate each value with a comma. Press Enter after you enter a boundary to see the placement of the boundary on the ruler.
   - To configure column boundaries using the ruler, use the mouse to drag the boundary to the appropriate location on the ruler for the first column. Each mark on the ruler represents a byte. To add new boundaries, click the ruler or click **Add**. Use the mouse to drag boundaries to the appropriate position on the ruler as necessary.

The following image shows a fixed-width file format with boundaries configured for six columns:



5. To add or change column names and edit datatypes, click **Edit Columns** and then enter the column name and select the datatype for each column.
6. To specify advanced properties, click **Additional Attributes** and specify the following properties:

Property	Description
Line sequential	Ends each row with a newline character. Must be enabled when you use a fixed-width file format for a flat file source or target object. Line sequential is enabled by default.
Number of rows to skip	Number of initial rows to skip. For example, you might want to skip blank rows or header rows.
Number of bytes to skip after column ending	Number of bytes between the last column of one row and the first column of the next.
Null character type	Whether the null character is text or ASCII.
Null character	Character to represent a null value.
Repeat null character	Reads repeat null characters in a single field as a single null value.

Property	Description
Strip trailing blanks	Removes trailing blanks from string values.
Default Date Format	Date format to use for the column when a date format is not specified in the flat file connection. To specify the default date format, enter the following information: - Whether the field width is adjusted or fixed. Enter A for adjusted width or F for fixed width followed by two spaces. - The field width in bytes. - The date format. For example, if you want the format to be fixed width with a width of 12 bytes and date format of DD/MM/YYYY, enter the following text: `F  12 DD/MM/YYYY`

To save the advanced properties, click **OK**.

7. To save the fixed-width file format, click **Save**.

CHAPTER 6

Hierarchical mappers

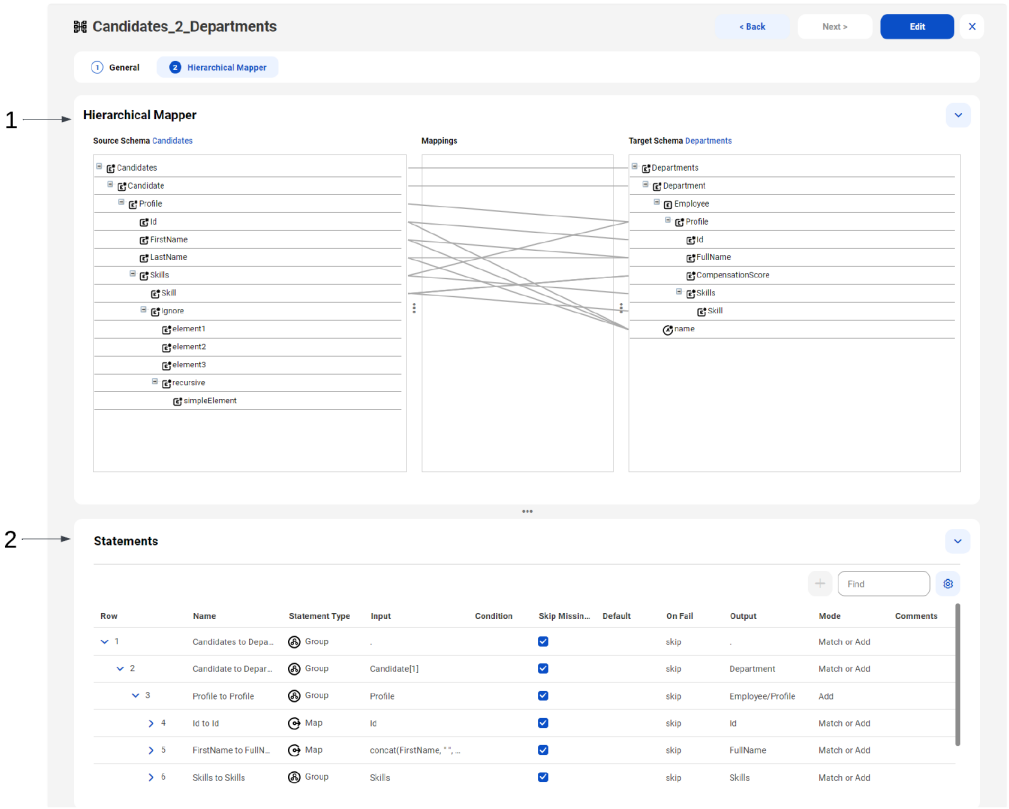
A hierarchical mapper maps a hierarchical schema, such as the schema for an industry-standard message, to another hierarchical schema. For example, you can map the XML output of a parser data service to a target XSD file.

A hierarchical mapper loads a source schema and a target schema. It uses statements to link source elements from the source schema to target elements in the target schema. To generate a statement, you can drag a source element to a target element, or you can manually create a statement. In the statement, you define how to convert the data from hierarchical input to hierarchical output as well as define conditions and filters.

For example, you can load a source schema that represents customer invoice data in XML format and map it to a target schema that represents customer orders in XML format. You can use statements to group customer orders by month and to map contact information and order totals.

You can use a hierarchical mapper in a Data Services transformation to process hierarchical input and convert it to hierarchical output that uses a different hierarchical structure, such as converting an XML document into a different XML document or converting a JSON document into a different JSON document.

The following image shows a hierarchical mapper:



1. Hierarchical mapper. The mapper displays the source and target schemas. You can drag a source element to a target element to generate a statement. You can also hover over an element to copy its XPath expression or view the element properties.
2. Statements. Edit statement properties or add a new statement.

To use a hierarchical mapper, perform the following tasks:

1. Optionally, create hierarchical schemas based on the hierarchical documents, such as XML or JSON documents. For more information, see Chapter 7, “Hierarchical schemas” on page 41.

   You don't need to create hierarchical schemas if you use data services from the data services repository to load the source and target schemas. The hierarchical mapper generates a hierarchical schema for the data service automatically.
2. Create a hierarchical mapper to map the source schema to the target schema.
3. Create a mapping and select the hierarchical mapper in a Data Services transformation to process the hierarchical input and convert it to hierarchical output. For more information about creating a mapping, see *Mappings*. For more information about the Data Services transformation, see *Transformations*.

Statements

A statement defines how hierarchical input in the source schema is mapped to hierarchical output in the target schema. When you drag a source element to a target element, the hierarchical mapper generates a statement that you can edit.

You can create a statement for each source element, or you can nest statements to make them dependent on other statements. A statement can be a parent to a group of child statements where each time the parent statement runs, the child statements run as well. Child statements appear indented below the parent statement.

You can create a statement by dragging a source element to a target element. The hierarchical mapper identifies the statement type, input, and output, generates a statement, and validates it. You can also manually add and configure statements. To delete a statement, click **Delete** or select the link between the source and target schema and press **Delete**.

Statement types

Each statement type represents the type of elements that are mapped between the source and target schemas. In the case of a Router statement, the statement type represents the action to take to map the elements.

You can define the following types of statements:

Map

A Map statement links a simple source element to a simple target element.

Group

A Group statement links a complex source element to a complex target element.

Repeating Group

A Repeating Group statement links a repeating complex source element to a repeating complex target element.

Router

A Router statement uses conditions to evaluate each source element and process it based on the input value.

Option

An Option statement defines a condition in a Router statement. If the condition evaluates to true, the hierarchical mapper runs the Option statement.

Default

A Default statement defines the default statement to run in a Router statement if none of the Option statements apply.

Map statement

A Map statement links a simple source element to a simple target element. The input must be a single value or a constant value. A Map statement can be a child of a Group, Repeating Group, or Option statement.

A Map statement can be a simple mapping between two elements such as a name element in the source and a name element in the target, or it can be complex mapping that uses expressions. For example, you want to map the Id, FirstName, and LastName elements in the source schema to the Name attribute in the target

schema. You can configure a Map statement with the input expression `concat(Profile/Id, "-", Profile/FirstName, "-", Profile/LastName)` and the output expression `@Name`.

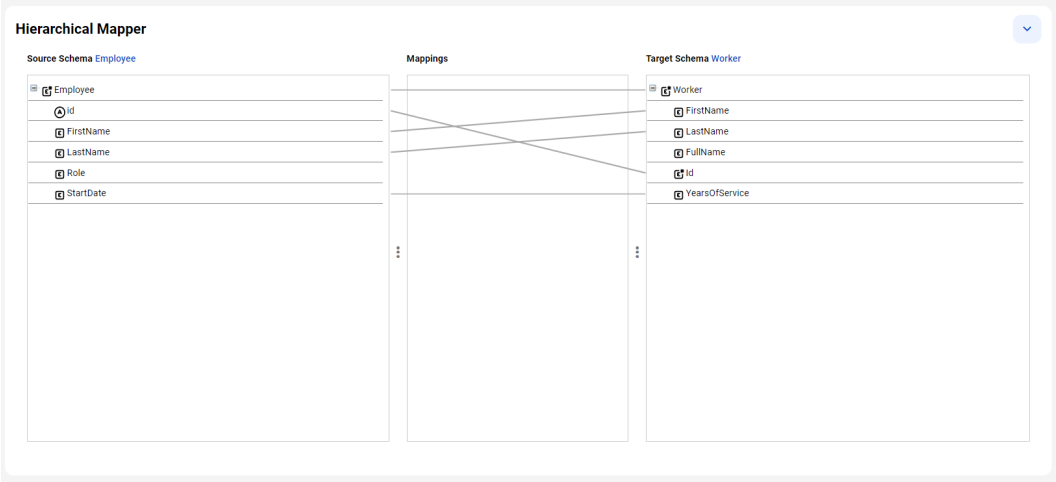
Group statement

A Group statement links a complex source element to a complex target element. You can nest other statement types as children of the Group statement.

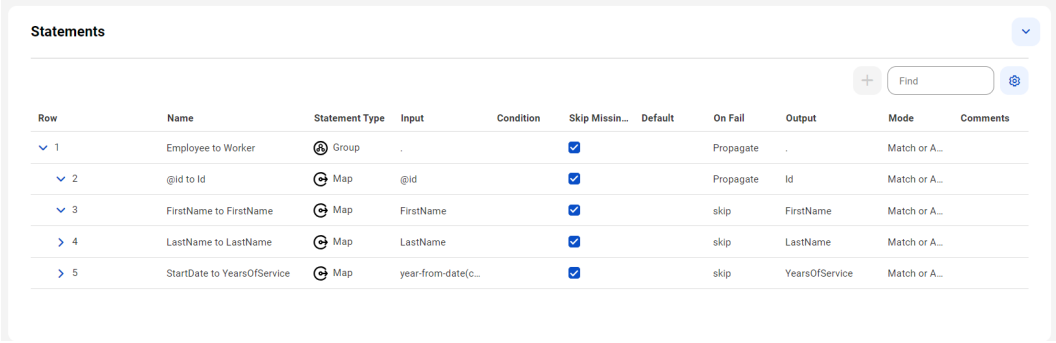
You can use a Group statement to provide a common context for all the child statements to pass or fail simultaneously. You can also use a Group statement to help you organize or simplify other statements.

For example, you want to map an Employee parent element in the source schema to a Worker parent element in the target schema where both parent elements contain nested child elements such as FirstName and LastName. You can create a Group statement to map the Employee parent element to the Worker parent element, and you can create nested Map statements to map the child elements of the Employee parent element to the child elements of the Worker parent element.

The following image shows the how the elements are mapped in a hierarchical mapper:



The following image shows the Group statement and the nested Map statements:



Repeating Group statement

A Repeating Group statement links a repeating complex source element to a repeating complex target element. The Repeating Group statement runs each time the source element appears in the source schema.

A Repeating Group statement contains iterated Map statements. The input to the Repeating Group is an expression that evaluates to a sequence of elements or values.

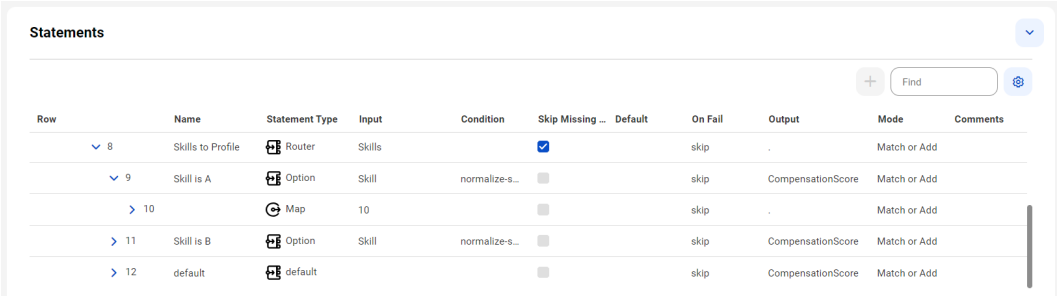
Router, Option, and Default statements

A Router statement uses conditions to evaluate each source element and process it based on the input value.

A Router statement contains one or more Option statements and one Default statement. When the hierarchical mapper processes a source element, it runs only the Option statement with a condition that matches the input value. If none of the conditions match the input value, the hierarchical mapper runs the Default statement.

For example, you're hiring employees for your organization and you want to map their skills to the compensation scores in their employee profiles. You can use a Router statement with Option statements that evaluate whether a candidate has Skill A or Skill B and assign the appropriate compensation score. If a candidate has neither skill, a Default statement runs to assign the compensation score.

The following image shows the Router statement and the nested Option and Default statements:



Statement properties

A hierarchical mapper automatically configures statement properties when it generates a statement, but you can edit the statement properties or manually configure a statement.

To add or remove rows, use the **Settings** button. You can also search for a statement using a string. When you select the row for a statement, the corresponding mapping is highlighted in the **Hierarchical Mapper** area. Similarly, when you select an element in the hierarchical mapper, the corresponding statement is highlighted in the **Statements** area.

The following table describes the statement properties:

Property	Description
Row	Row number to identify the statement. The row number is generated automatically.
Name	Name to identify the statement. You can change the name at any time, and the statement name doesn't need to be unique. You can use the statement name to help you find the statement in the session log.
Statement Type	Type that defines how the statement runs when a source element appears in the source schema.
Input	XPath or expression consisting of XPaths that defines the input element in the source schema. The expression can evaluate to an element or to a value.
Condition	XPath or expression consisting of XPaths that defines a condition to map the source element to the target element. For example, `normalize-space(string(.)) = 'A'`

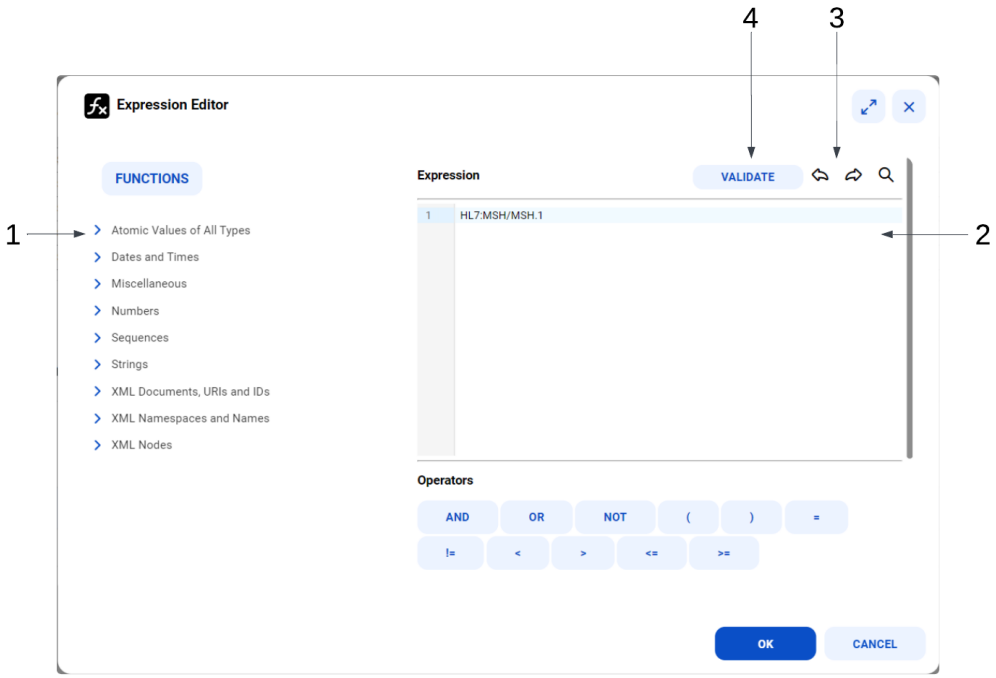
Property	Description
Skip Missing Input	Determines whether to skip the mapping statement if the input expression doesn't match an element in the source schema. Choose one of the following options: - Enabled. If the source element doesn't exist, the hierarchical mapper skips the statement without error. - Disabled. If the source element doesn't exist, the statement fails.
Default	Default value to set the value of the target element if the hierarchical mapper can't find the source element.
On Fail	Determines what to do if the statement fails. Choose one of the following options: - Skip Statement. If the statement fails, skip the statement. - Skip Iteration. If the statement fails and it's part of a group, skip every iteration of the statement. - Propagate. If the statement fails, the parent statement also fails.
Output	XPath or expression consisting of XPaths that defines the output element in the target schema.
Mode	Determines whether the hierarchical mapper adds an output element to the target schema or matches the value of the output expression to an existing element in the target schema. Choose one of the following options: - Add. Creates an element in the target schema. If the element is not a multiple-occurring element and the element already exists in the target schema, the statement fails. - Match. Matches the value of the output expression to an existing element in the target schema. If the element doesn't exist, the statement fails. - Match or Add. If the value of the output expression exists in the target schema, the hierarchical mapper matches the output to the target element. If the element doesn't exist in the target schema, the hierarchical mapper creates an element in the target schema.
Comments	Comments that you want to make about the statement.

XPath expression editor

Create input, condition, and output expressions in the XPath expression editor. XPath is a query language that performs computations and arithmetic calculations on hierarchical elements.

You can use XPath expressions to define the context for a statement or to define conditions to transform and filter the data. The function library in the expression editor is standard for the W3C XML Path Language.

When you click the **Expression Editor** button in the input, condition, or output property of a statement, the expression editor appears. The following image shows the expression editor:



1. Add functions to the expression.
2. Edit the expression.
3. Undo and redo edits to the expression.
4. Validate the expression.

You can add functions to the expression by clicking **Add** next to the function that you want to use. You can also type in the expression manually. To validate the expression, click **Validate**.

Creating a hierarchical mapper

Create a hierarchical mapper to map a hierarchical schema, such as the schema for an industry-standard message, to another hierarchical schema.

1. Click **New** > **Components** > **Hierarchical Mapper** and then click **Create**.
2. In the **Hierarchical Mapper** designer, configure the properties on the **General** page.

The following table describes the general properties:

Property	Description
Name	Name of the hierarchical mapper.
Description	Description of the hierarchical mapper.
Location	Project folder where you want to save the hierarchical mapper.

3. Click **Next**.
4. On the **Hierarchical Mapper** page, click **Select a source schema**. Use one of the following options based on the source type that you select:
   - If you select **Hierarchical Schema**, click **Browse** to browse for the source schema that you want to load into the designer, and then click **Select**.
   - If you select **Data Services Repository**, click **Browse** to browse for the data service and click **Select**. Then, select the root that you want to load into the designer.
5. Click **Select a target schema**. Use one of the following options based on the target type that you select:
   - If you select **Hierarchical Schema**, click **Browse** to browse for the target schema that you want to load into the designer, and then click **Select**.
   - If you select **Data Services Repository**, click **Browse** to browse for the data service and click **Select**. Then, select the root that you want to load into the designer.
6. To generate mapping statements, use one of the following options:
   - Drag a source element to a target element to automatically generate a statement.
   - Scroll down to the list of statements and click the **Add** button to add a new statement and configure the statement properties.
7. After you configure the statements, click **Save**.

CHAPTER 7

Hierarchical schemas

A hierarchical schema is an asset that defines the expected hierarchy of data. The schema is based on a sample or schema file that you upload in Data Integration.

You can use a hierarchical schema in a Hierarchy Builder transformation or a Hierarchy Parser transformation. You can create the hierarchical schema as a standalone asset or within the transformation. Whether you create the schema as a standalone asset or within a transformation, you can associate it with any transformation.

After you create a hierarchical schema, you can edit it to change the root or the schema definition. You can't edit or delete a hierarchical schema if it's used in a transformation.

Choosing a sample or schema file

When you create a hierarchical schema, you base it on either a sample file or a schema file. To improve accuracy, use a schema file instead of a sample file whenever possible.

Schema files

When you use a schema file, consider the following rules and guidelines:

- The schema can't contain recursive elements.
- Each schema file can contain a maximum of 10,000 elements. To use a schema that contains more than 10,000 elements, split it into multiple files.
- To use a schema that references other schemas, you can upload each file individually or upload them together in a ZIP file. If the ZIP file contains other files in addition to XSD files, Data Integration ignores the additional files and uses only the XSD files to generate the hierarchical schema.
- If you need to upload multiple schema files with the same name, upload them in a ZIP file. If you upload them individually, Data Integration renames the duplicate files.

Sample files

When you use a sample file, ensure that it represents the data that you expect to process, including all possible fields, all permutations of the values and data types, and values that represent the length of the data that the mapping will process.

Creating a hierarchical schema

To create a hierarchical schema, define general properties and upload a sample or schema file.

1. Click **New** > **Components** > **Hierarchical Schema**.
2. On the **New Hierarchical Schema** page, specify a name and location for the hierarchical schema, or use the default values.
3. Optionally, enter a description of the hierarchical schema.
4. Click **Upload** to select a file as the basis of the hierarchical schema.
5. In the **Upload Schema/Sample File** dialog box, click **Choose File** to browse for a file, and then click **OK** to upload it.
6. If you upload an XSD file that references other XSD files, Data Integration displays a list of the referenced files. Click **Upload** to upload each referenced file.
7. If you upload an XSD file with multiple possible root elements, select a root element from the drop-down menu.
8. To save the hierarchical schema, click **OK**.

CHAPTER 8

Industry data service customizer

Create an industry data service customizer to customize an industry-standard message and publish it as a custom data service to the data services repository.

A customizer provides out-of-the-box message types based on the messaging standard. You select a message type and customize the message structure by adding, editing, and removing elements. You can save the message to continue editing it later, or you can publish the message as a data service so you can use it in a Data Services transformation. You can't edit the published data service, but you can edit the customizer and republish the message as another data service using a different name.

Message definition

The message definition includes the message structure, and each element in the message structure contains properties that you can edit. The properties depend on the element and its position in the message structure.

Message structures are defined according to the messaging standard. For detailed information about the message structure and properties, see the appropriate messaging standard. For example, for detailed information about HL7 message structure and properties, see the HL7 documentation.

Message structure

The message uses a hierarchical structure that labels each structural element with an icon.

The following table describes the icons in the message structure:

Icon	Structural element	Description
	Segment group	A group of segments that can be repeated.
	Segment	A sequence of data elements.
	Composite or field	A structure of data elements.

Icon	Structural element	Description
	Component	A container for a data element.
	Data element	A simple data item.

Messaging standards use different names for structural elements and define how they are nested. In some standards, elements have names such as record and field instead of segment and composite. A segment might contain composites and data elements, or a composite might contain data elements. Some standards define additional constructs such as options and alternatives.

Global and positional settings

Structural elements have global settings and positional settings that you can edit. Global settings apply to all instances of the element, and positional settings apply to the selected instance of the element.

For example, a global setting might allow an element to contain any of 10 enumerated values. A positional setting might limit the element to a subset of two values.

Customizing an industry-standard message

Customize an industry-standard message to add, edit, and remove elements in the message structure and publish the message as a data service.

1. In Data Integration, click **New** > **Components** > **Industry Data Services Customizer**.
2. Click **Create**.
3. Configure the general properties.

   The following table describes the general properties:

Property	Description
Name	Name of the message.
Location	Location of the message. Browse to a folder where you want to store the message or use the default location.
Description	Description of the message.
Message Type	Out-of-the-box message type based on the messaging standard.

4. Configure the message structure and edit element properties.
   - To add an element, see "Adding an element to the message structure" on page 45.
   - To edit element properties, see "Editing element properties" on page 45.

- To add enumerations, see "Adding enumerations" on page 46.
- To delete an element, see "Deleting an element from the message structure" on page 46.

5. Click **Save**.
6. To publish the message as a data service, click **Publish**.
7. Configure the data service properties.

   The following table describes the data service properties:

Property	Description
Service Name	Name of the data service.
Runtime Environment	Runtime environment that contains a running Secure Agent.
Publish Options	Defines how to publish the data service for consumption. Select one or more of the following options: - Parser. The data service is used to receive messages from your partners. - Serializer. The data service is used to send messages to your partners. - Restricted Serializer. The data service is used to send messages to your partners, including validation messages. Applies to HIPAA messages. The customizer will publish a data service for each option that you select.

8. Click **Publish**.

After you publish the message as a data service, you can view the data service in the data services repository in Administrator and use it in a Data Services transformation.

Adding an element to the message structure

Use the customizer to add an element to the message structure. When you add an element, the customizer populates the element with nested elements based on the messaging standard.

1. In the message structure, select the element.
2. Click the **Add** button and select one of the following options:
   - **New**. Append an element to the end of the list of elements.
   - **Insert Within**. Insert the element within the selected element.
3. Perform one of the following tasks:
   - Select an existing element to add.
   - Create a new element by specifying the element ID.
4. Click **Select** or **Create**, respectively.

Editing element properties

Use the customizer to edit element properties in the message structure.

1. In the message structure, select the element.

   The **Properties** panel displays the element properties.
2. In the **Properties** panel, click the property value and enter a new value.
3. Optionally, hover over the property and select **Restore default value** to undo changes.

Adding enumerations

Add enumerations to an element to create a table of permitted values.

1. In the message structure, select the element.

   The **Properties** panel displays the element properties.
2. In the **Properties** panel, hover over the **Enumerations** row and select the **Update Enumeration** icon.
3. Specify a name for the enumerations table.
4. Click the **Add** icon.
5. Enter a value and description for the enumeration.
6. Click **Update**.

Deleting an element from the message structure

Use the customizer to delete an element from the message structure.

1. In the message structure, select the element.
2. Click the **Delete** button.
3. Click **OK**.

Message properties

Messaging standards define the elements and properties that you can configure in the message structure. This section describes the properties for each standard.

HIPAA message properties

HIPAA is an industry standard for administrative and financial health-care transactions. HIPAA is based on the X12 standard.

HIPAA messages use transaction sets, loops, segments, composites, and data elements.

Transaction set

The following table describes the transaction set properties:

Property	Setting Type	Description
ID	Global	Message identifier.
Library type	Global	Messaging standard. Set to HIPAA.
Library version	Global	Version of the messaging standard.
Industry ID	Global	Industry identifier.

Property	Setting Type	Description
Standard name	Global	Message name.
Functional group	Global	The group that the message belongs to.

Loop

The following table describes the loop properties:

Property	Setting Type	Description
ID	Global	Loop identifier.
Repetitions	Positional	Number of times that the loop is repeated such as 1 or 2. For an unlimited number, enter `>1`.

Segment

The following table describes the segment properties:

Property	Setting Type	Description
ID	Global	Segment identifier.
Standard name	Global	Segment name.
Standard qualifier	Global	Index of the data element that qualifies the segment.
Usage	Positional	Required, optional, conditional, or not used.
Repetitions	Positional	Number of times that the segment is repeated such as 1 or 2. For an unlimited number, enter `>1`.
Industry name	Positional	Industry-standard descriptive name of the segment.

Composite

The following table describes the composite properties:

Property	Setting Type	Description
ID	Global	Composite identifier.
Standard name	Global	Composite name.
Usage	Positional	Required, optional, conditional, or not used.
Repetitions	Positional	Number of times that the composite is repeated such as 1 or 2. For an unlimited number, enter `>1`.

Data element

The following table describes the data element properties:

Property	Setting Type	Description
ID	Global	Data element identifier.
Standard name	Global	Data element name.
Type	Global	Data element type.
Min length	Global	Minimum length of the data.
Max length	Global	Maximum length of the data.
Usage	Positional	Required, optional, conditional, or not used.
Repetitions	Positional	Number of times that the data element is repeated such as 1 or 2.
Valid codes	Positional	A list of valid codes.

HL7 message properties

The HL7 messaging standard implements the Health Level Seven version 2.x messaging standard used in the health services industry. The HL7 standard is used world-wide in hospital and medical information systems.

HL7 messages use segment groups, segments, fields, components, and types.

Message

The following table describes the message properties:

Property	Setting Type	Description
ID	Global	Message identifier.
Library type	Global	Messaging standard. Set to HL7.
Library version	Global	Version of the messaging standard.
Description	Global	Description of the message.
Category	Global	Category that the message belongs to, such as patient referral.

Segment group

The following table describes the segment group properties:

Property	Setting Type	Description
ID	Global	Segment group identifier.
Usage	Positional	Mandatory or optional.
Repetitions	Positional	Number of times that the segment is repeated such as 1 or 2. For an unlimited number, enter `>1`.

Segment

The following table describes the segment properties:

Property	Setting Type	Description
ID	Global	Segment identifier.
Description	Global	Description of the segment.
Usage	Positional	Mandatory or optional.
Repetitions	Positional	Number of times that the segment is repeated such as 1 or 2. For an unlimited number, enter `>1`.

Field

The following table describes the field properties:

Property	Setting Type	Description
ID	Global	Field identifier. The identifier is usually the ID of the parent element followed by an index number.
Description	Global	Description of the field.
Type	Global	Field type.
Max Length	Global	Maximum length of the data.
Enumerations	Global	A list of permitted data values.
Usage	Positional	Mandatory or optional.
Repetitions	Positional	Number of times that the segment is repeated such as 1 or 2. For an unlimited number, enter `>1`.

Component

The following table describes the component properties:

Property	Setting Type	Description
ID	Global	Component identifier. The identifier is usually the ID of the parent element followed by an index number.
Description	Global	Description of the component.
Type	Global	Component type.
Usage	Global	Mandatory or optional.

Type

The following table describes the type properties:

Property	Setting Type	Description
ID	Global	Type identifier.
Description	Global	Description of the type.

CHAPTER 9

Intelligent structure models

A CLAIRE® intelligent structure model is an asset that Intelligent Structure Discovery creates based on input that represents the data that you expect the model to parse at run time.

Intelligent Structure Discovery determines the underlying patterns and structures of the input that you provide for the model and creates a model that can be used to transform, parse, and generate output groups.

Long, complex files with little or no structure can be difficult to parse. Intelligent Structure Discovery can automatically decipher input data and discover the patterns, repetitions, relationships, and types of data in unstructured files.

After intelligent structure discovers the structure of the data, you can refine and test the structure, and then save or export it. When you save or export an intelligent structure, Intelligent Structure Discovery creates an intelligent structure model in an `.amodel` file.

Intelligent Structure Discovery creates a model that expresses the expected output data. You can use an intelligent structure model in mappings to parse unstructured, semi-structured, or structured data.

You can create models from the following input types:

- Text files, including delimited files such as CSV files and complex files that contain textual hierarchies
- Machine generated files such as weblogs and clickstreams
- JSON files
- XML files
- ORC files
- Avro files
- Parquet files
- Microsoft Excel files
- Data within PDF form fields
- Data within Microsoft Word tables
- XSD files
- Cobol copybooks

Using intelligent structure models in mappings

To use an intelligent structure model in a mapping, add a Structure Parser transformation or a Hierarchy Builder transformation to the mapping.

An intelligent structure model is required for Structure Parser transformations and is optional for Hierarchy Builder transformations.

When you configure the transformation, select or create an intelligent structure model, select the type of input that the transformation expects to receive, and select the output that you pass to downstream transformations. The model can generate relational, XML, JSON, JSON Lines, or Hadoop format output.

Note: If the output contains data of type DATE, Intelligent Structure Discovery represents the data in the mapping as String, not as Date or DateTime.

Models that you use with a Structure Parser transformation in a mapping can contain up to 12,000 fields. The Structure Parser transformation might fail to load models that contain more than 12,000 fields.

Tip: When you create a model that isn't JSON, XML, or XSD-based, and the model contains nested repeating groups, you can reduce the number of fields in the model by normalizing the output data. Intelligent Structure Discovery normalizes the input data by default for JSON, XML, and XSD-based models that contain nested repeating groups.

For more information about the Structure Parser and Hierarchy Builder transformations, see *Transformations*.

Using intelligent structure models in mappings in advanced mode

To use an intelligent structure model in a mapping in advanced mode, configure the Source transformation to discover the data structure at the source or add a Structure Parser transformation downstream in the mapping.

The model generates hierarchical output. Models that you use in advanced mode can contain up to 8,000 fields.

To use an intelligent structure model when you configure the Source transformation, select an existing model or create a new model when you configure the Source transformation. You can use an intelligent structure model with an Amazon S3 V2 or a Microsoft Azure Data Lake Gen2 connection. The following table describes the actions that you perform when you configure the source properties to use an intelligent structure model:

Property	Action
Connection	Select an Amazon S3 V2 or a Microsoft Azure Data Lake Gen2 connection.
Source Type	Select **Single Object**.
Object	Select a file or folder as the source object: - To select a file, select **File** as the source type in the **Advanced** area, and then select a file of a type that is supported by Intelligent Structure Discovery. - To select a folder, select **Directory** as the source type in the **Advanced** area, and then select a folder that contains one or more Avro, Parquet, or ORC files.

Property	Action
Format	Select **Discover Structure**.
Intelligent Structure Model	Select one of the following options to associate a model with the transformation: - Select. Select an existing model. - New. Create a new model. Select **Design New** to create the model. Select **Auto-generate from sample file** for Intelligent Structure Discovery to generate a model based on sample input that you select.

For more information, see *Transformations*.

Using intelligent structure models in data engineering mappings

To use an intelligent structure model in a data engineering mapping, add the model to a data object. The model generates HTYPE output.

You can add an intelligent structure model to data objects, and then incorporate them into mappings in data engineering. To use an intelligent structure model in a data object, first export it from Data Integration to your local drive.

Use Informatica Developer to add the intelligent structure to a complex file data object, Amazon S3 data object, or Microsoft Azure Blob data object. You can add the data object to a data engineering mapping and process data on the Spark engine. For more information, see the *Data Engineering Integration User Guide*.

Using intelligent structure models in B2B Gateway inbound partner flows

To use an intelligent structure model in a B2B Gateway inbound partner flow, select an intelligent structure when you create the partner.

You can use an intelligent structure model to receive incoming Excel, TXT, and CSV files. Intelligent Structure Discovery writes the files to a CSV interface file on the B2B Gateway document store.

For more information, see the B2B Gateway help.

Intelligent Structure Discovery process

You can create an intelligent structure model using Intelligent Structure Discovery.

After you provide an input file, Intelligent Structure Discovery determines the underlying and repeating patterns of the data and creates a structure that represents the fields of data and their relationships. You can quickly model data for files whose structure is very hard, time consuming, and costly to find, such as log files,

clickstreams, customer web access, error text files, or other internet, sensor, or device data that does not follow industry standards.

The following image shows the process by which Intelligent Structure Discovery deciphers the underlying patterns of data and creates a model of the data patterns:



When the model is based on input from a Hadoop Files source, Intelligent Structure Discovery determines the data types to use in the model based on the types and structures in the Hadoop schema. In the model, Intelligent Structure Discovery creates an array of structures of key-value pairs for data of type map. Nodes of type enum are represented as strings.

Inputs for intelligent structure models

The input that you base an intelligent structure model on can be a sample file, an XSD schema, an Avro schema, or a Cobol copybook, based on the input that you expect to use the model for at run time.

Input files can be up to 1 MB in size. An input file can contain up to 30,000 simple fields. If the file contains more than 30,000 simple fields, Intelligent Structure Discovery creates the model without groups and ports. The number of levels in the hierarchy isn't limited.

To achieve optimal parsing results, ensure that the input that you provide when you create the model is broad enough and covers all the data elements that you expect the model to receive at run time. If the input is too limited, the parsing output will include unidentified data. If the input contains rows, it must contain at least three lines of data.

Use simplified input to generate the model. For example, if the input data has tables, provide a table with just a few sample rows rather than many rows of data. If you use a JSON input file that contains repeating groups of data, limit the number of repetitions.

If the model does not match the runtime input data, or only partially matches the input data, there might be a large amount of unidentified data and data loss. However, some variations will still be parsed.

Verify that the length of each combination of group name and field name doesn't exceed 80 characters. For example, if the name of the group that a field belongs to is `group`, the field name must not exceed 75 characters. If a combination of group name and field name exceeds 80 characters, mappings that use the model fail to run.

Discover structure from an entire XML or JSON sample file

By default, Intelligent Structure Discovery discovers the structure of the data based on the first portion of the input file. When you base a model on an XML or JSON file, you can choose to discover the structure of the data based on the entire file, up to 30 MB. Use this option if the first portion of the file doesn't represent all the input that you expect to use the model for at run time.

Note that when Intelligent Structure Discovery discovers the structure of the data based on the entire file, the discovery process might take a few minutes to complete.

Discover structure from a Microsoft Excel file

You can base a model on the following types of Microsoft Excel files: xla, xlam, xls, xlsm, xlsx, xlt, xltm, and xltx.

Using ORC files

You can use the model to read ORC files through a flat file connection in Data Integration. You can't use the model for ORC streaming.

Using multiple sample files in a model

After you create a model based on a JSON, XML, ORC, AVRO, or PARQUET sample file, you can use additional sample files to enrich the structure with fields that exist in the new samples. The additional files must be of the same file type as the type of file that the model is based on.

Using multi-file XSD schemas

Consider the following guidelines when you use an XSD schema that contains multiple XSD files as the model input:

- The schema files must be compressed.
- If the XSD files reside in a directory structure, to preserve the structure, the parent directory must be compressed.

Discover structure from large XSD schemas

When you base a model on an XSD schema, by default, Intelligent Structure Discovery can discover the structure of the data from schemas that are up to 1.5 MB in size. To use a larger file, perform one of the following actions:

- Choose to discover the structure of the data based on the entire schema, up to 30 MB.
- Zip the schema file and select the zip file as the input for the model.

If you use a large schema without taking one of these actions, Intelligent Structure Discovery treats the input as XML and discovers the structure based on partial data.

Using XML sample files in XSD-based models

When you create an XSD-based model to use in a Structure Parser transformation, you can attach an XML sample file to the model. The names and contents of the groups in the model appear in the Intelligent Structure Model page. When you associate the model with the Structure Parser transformation, use this information to decide which group to connect to the target. Attaching a sample file to the model doesn't affect or change the structure of the model.

Parsing JSON-encoded Avro messages

You can use models that are based on an Avro schema to parse JSON-encoded Avro messages.

Character encoding in XSD schemas

XSD schemas that you use as model inputs can use either UTF-8 or UTF-16 character encoding.

Output group definition

When you base an intelligent structure model on an XML or XSD file, you can choose how Intelligent Structure Discovery defines the output groups.

Before you discover the structure of your input file, you can choose one of the following options on the **Settings** tab:

Default output groups

Intelligent Structure Discovery can generate normalized or denormalized output groups. Intelligent Structure Discovery also generates groups based on XSD compositors, such as choice groups, if the input file uses them.

Normalized hierarchy relationships

This method generates output groups for each repeating element. Using normalized hierarchy relationships is generally faster and uses less memory than using default output groups.

Entity relationships

This method generates output groups for each repeating element and for complex data types. Using entity relationships creates the most compact representation of complex data types.

When Intelligent Structure Discovery generates entity relationships, it uses the following rules:

- Intelligent Structure Discovery adds a primary key with the name `<group name>_PK` to every output group.
- Intelligent Structure Discovery generates an output group for the following global elements:
  - The root element
  - Elements that are part of a recursive reference
  - Repeated elements with the indicator `maxOccurs = "unbounded"`
- Intelligent Structure Discovery adds foreign keys to output groups based on global elements for each output group that refers to the global element. The foreign keys use the name `<parent group name>_FK`. If there are multiple parent groups that refer to the global element, the output group for the global element has multiple foreign keys.
- Intelligent Structure Discovery generates an output group for every complex type. The name of the output group is the name of the complex type.
- If a complex type has a one-to-one relationship with its parent element, Intelligent Structure Discovery adds a foreign key with the name `<parent element name>_<type name>_FK` to the output group of the complex type.
- If there is a one-to-many or many-to-many relationship between a complex type and its parent elements, Intelligent Structure Discovery generates a bridge table called `<group name 1>_<group name 2>`. The bridge table includes only a primary key, a foreign key to the first output group, and a foreign key to the second output group.

In every output group, the first field is the generated primary key, followed by any foreign keys, and then the fields present in the schema.

Simplify Excel

In Intelligent Structure Discovery, you can use the Simplify Excel mode to parse Excel workbooks.

Simplify Excel spreadsheets for better performance and to consume less heap memory and CPU cores.

Consider the following rules and guidelines when you simplify spreadsheets:

- Simplify Excel mode doesn't detect merged cells in a spreadsheet.
- When you parse spreadsheets containing more than one table, the intelligent structure model doesn't detect the tables as separate tables.
- When each cell in a column contains data separated by a hyphen, such as 12-654, the intelligent structure model doesn't recognize the data as two separate columns.
- When each cell in a column contains a number that contains two hyphens, such as 12-345-678, the intelligent structure model doesn't identify it as three separate rows.
- You can't write Unassigned data to a different target.
- You can't parse data in cells that use the Advanced number format.
- In the Structure Parser transformation, when you use Simplify Excel mode, you can only use relational output.

Repeating groups

Intelligent Structure Discovery creates a repeating group for input data such as a table or array, where a row or set of fields repeats as a group.

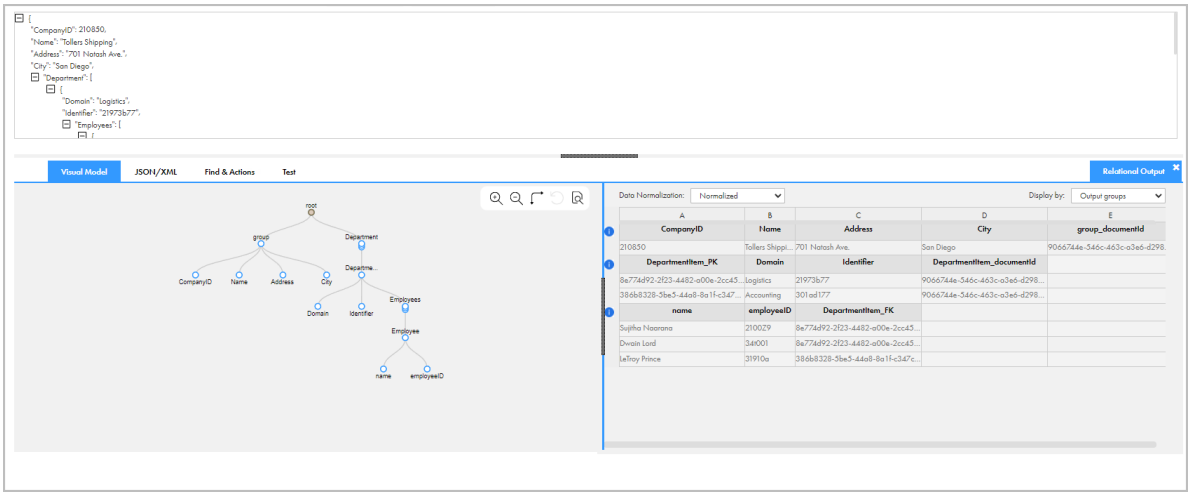
When a model that is based on a JSON, XML, or XSD file contains nested repeating groups, that is, repeating groups within repeating groups, Intelligent Structure Discovery assigns each nested repeating group to its own output group. For models that are based on other input types, you can manually assign nested repeating groups to their output groups. You can assign all nested repeating groups to their output groups by normalizing the data in the model, or you can assign individual nested repeating groups to their output groups by promoting them. Assigning nested repeating groups to their output groups reduces the number of ports in the model.

For example, you create a model based on the following JSON input:

```
{
  "CompanyID": 210850,
  "Name": "Tollers Shipping",
  "Address": "701 Natash Ave.",
  "City": "San Diego",
  "Department": [
    {
      "Domain": "Logistics",
      "Identifier": "21973b77",
      "Employees": [
        {
          "name": "Sujitha Naarana",
          "employeeID": "2100Z9"
        },
        {
          "name": "Dwain Lord",
          "employeeID": "34t001"
        }
      ],
    },
```

```
        {
      "Domain": "Accounting",
      "Identifier": "301ad177",
      "Employees": [
        {
          "name": "LeTroy Prince",
          "employeeID": "31910a"
        }
      ]
    }
  ]
}
```

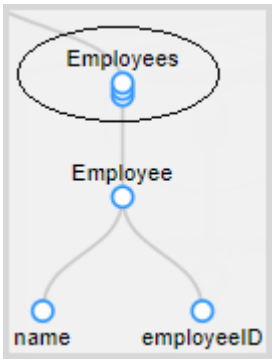
The following image shows the structure that Intelligent Structure Discovery discovers:



The structure contains a nested repeating group. The top level repeating group is **Department**. Each **Department** group contains an **Employees** repeating group that contains an **Employee** group node. The **Employee** group node contains **name** and **employeeID** child nodes.

A repeating group is displayed in the Visual Model with three overlapping circles for the parent node.

The following image shows the **Employees** repeating group:



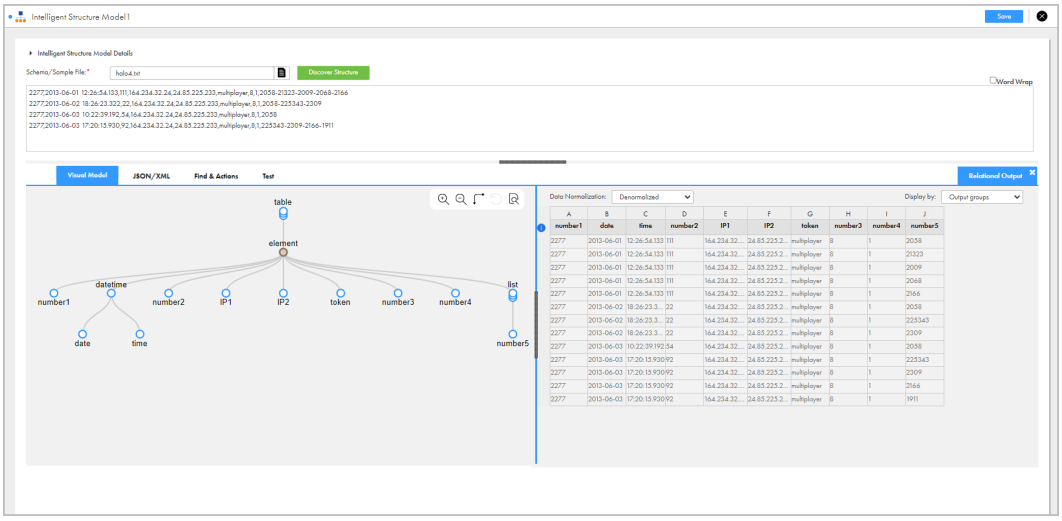
Primary and foreign keys

For models that contain nested repeating groups, Intelligent Structure Discovery uses primary and foreign keys to identify the relationships between repeating groups and their child nodes.

When a model that is based on a JSON, XML, or XSD file contains nested repeating groups, Intelligent Structure Discovery can normalize the output data and assign each nested repeating group to its own output group. For models that are based on other input types, you can manually assign nested repeating groups to their output groups.

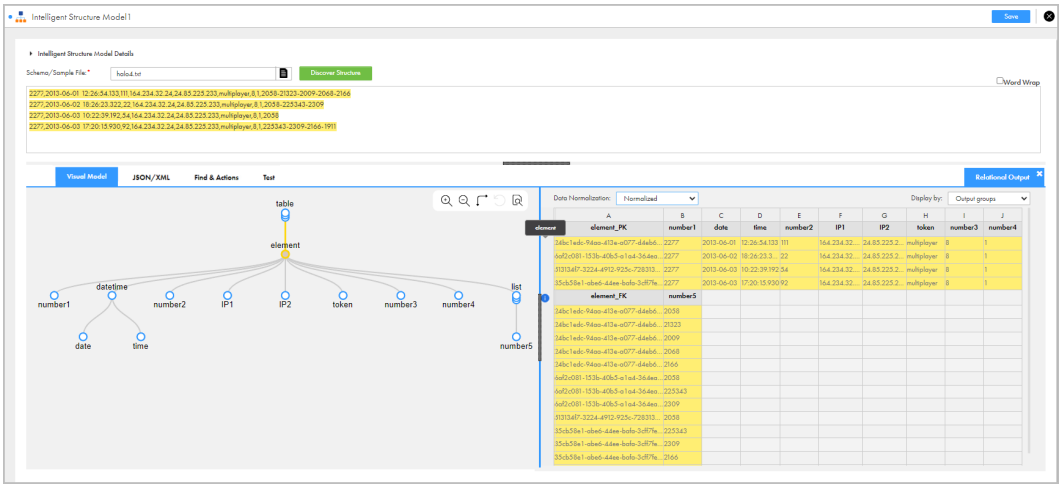
When a nested repeating group is assigned to its output group, Intelligent Structure Discovery adds a primary key to the parent group and a foreign key to the child group.

The following image shows the structure that Intelligent Structure Discovery discovered from a CSV input file:



In this model, the **list** group is part of the **element** output group. The data normalization mode is denormalized, and the **list** nested repeating group isn't assigned to a separate output group.

The following image shows the same model after you change the data normalization mode to normalized:



Intelligent Structure Discovery generates two separate output groups, the **element** output group, and the **list** output group. You can view a group name by hovering over the tip icon to the left of the group.

Intelligent Structure Discovery added the primary key **element_PK** to the parent **element** output group, and the foreign key **element_FK** to the nested **list** output group.

You can select a different node as the primary key by defining it as a record ID. When you change the record ID, Intelligent Structure Discovery creates a corresponding foreign key in the nested group.

Data drifts

Intelligent structure models can address data drift in certain cases, as in the following example. In this example, the sample data used to create the model contains the following text:

```
05967|2014-09-19|04:49:50.476|51.88.6.206|custid=83834785|cntry=Tanzania|city=Mtwango|
movie={b1027374-6eec-4568-8af6-6c037d828c66|"Touch of Evil"}|paid=true
01357|2014-11-13|18:07:57.441|88.2.218.236|custid=41834772|movie={01924cd3-87f4-4492-
b26c-268342e87eaf|"The Good, the Bad and the Ugly"}|paid=true
00873|2014-06-14|09:16:14.522|134.254.152.84|custid=58770178|movie={cd381236-53bd-4119-
b2ce-315dae932782|"Donnie Darko"}|paid=true
02112|2015-01-29|20:40:37.210|105.107.203.34|custid=49774177|cntry=Colombia|city=Palmito|
movie={ba1c48ed-d9ac-4bcb-be5d-cf3afbb61f04|"Lagaan: Once Upon a Time in India"}|
paid=false
00408|2014-06-24|03:44:33.612|172.149.175.30|custid=29613035|cntry=Iran|city=Bastak|
movie={3d022c51-f87f-487a-bc7f-1b9e5d138791|"The Shining"}|paid=false
03568|2015-01-07|11:36:50.52|82.81.202.22|custid=27515249|cntry=Philippines|
city=Magallanes|movie={ad3ae2b4-496e-4f79-a6dd-202ec932e0ae|"Inglourious Basterds"}|
paid=true
```

The input data that the model parses contains the following text:

```
0448|2015-04-07|01:50:5.35|27.248.247.174|custid=613068|cntry=Iran|city=SarÄ•b|
movie={50fb37b-621-484e-a565-2b5c1cbdc43|"Network"}|paid=false|ua=Mozilla/5.0 (Windows
NT 5.1)
02780|2014-12-28|08:14:58.685|17.2.236.233|custid=731|cntry=Greece|city=NÃ©a RÃ³da|
movie={1876aea0-3cb5-4c7a-22f-d33f233210|"Full Metal Jacket"}|paid=true|ua=Mozilla/5.0
(Macintosh; Intel Mac OS X 10_10_1)
03353|2015-04-20|21:02:40.532|143.48.11.171|custid=83736441|cntry=Russia|city=Mozhaysk|
movie={67272f85-bfc-418a-82ea-a7c4ae6b028a|"Gangs of Wasseypur"}|paid=true|
ua=Mozilla/5.0 (iPad; CPU OS 5_1 like Mac OS X)
04073|2014-10-25|15:33:03.442|87.235.48.100|custid=861028|cntry=Indonesia|city=Lamalera|
movie={4a511f3-6367-4017-874e-50a46f5ea567|"Shutter Island"}|paid=false|ua=Mozilla/5.0
(X11; Linux x86_64)
02170|2015-02-1|23:36:40.271|25.14.204.46|custid=1240203|cntry=Albania|city=LukovÃ«|
movie={2047efa-22c6-431c-87d4-ca73af1034|"The Grapes of Wrath"}|paid=false|
ua=Mozilla/5.0 (Windows NT 6.1)
```

The input data contains additional data that isn't defined in the model. However, the model can parse this variation.

Unassigned Data

When Intelligent Structure Discovery discovers the structure of a file, it might create a field called *Unassigned Data* if the file has records that don't match the intelligent structure model.

Intelligent Structure Discovery creates the *Unassigned Data* field for a variety of reasons, including the following scenarios:

- A delimited file, such as a CSV or log file, contains more elements than expected and Intelligent Structure Discovery can't parse the data drift.

- A record in a JSON file isn't in the intelligent structure model or exceeds the maximum record size.

Maximum record size

When Intelligent Structure Discovery discovers the structure of a JSON sample file, it uses the maximum record size to identify repeating records. If a record is larger than the maximum record size, Intelligent Structure Discovery assigns the record to the *Unassigned Data* field.

The default maximum record size is 640,000 bytes. You can increase the maximum record size to avoid the use of the *Unassigned Data* field.

To edit the maximum record size, use Administrator to configure a JVM option in the Data Integration Server properties. Use the following syntax to define the maximum record size:

```
-DISD_MAX_RECORD_SIZE=<size in bytes>
```

For example, to define a maximum record size of 2 MB, enter the following value for the `JVMOption1` property:

```
-DISD_MAX_RECORD_SIZE=2000000
```

Note: Increasing the maximum record size increases the memory consumption of the discovery process. Therefore, you might need to perform one or both of the following actions:

- Increase the maximum JVM heap size in the Data Integration Server properties. To increase the JVM heap size, set one of the JVM options to `-Xmx<heap size in megabytes>`.
- Increase the memory of the Secure Agent machine. If the Secure Agent runs on an Informatica Cloud Hosted Agent, contact Informatica Global Customer Support.

For more information about configuring Data Integration Server properties and the Secure Agent, see the Administrator help.

Creating an intelligent structure model

Create an intelligent structure model based on input that represents the data that you expect the model to parse at run time.

1. Click **New** > **Components** > **Intelligent Structure Model**, and then click **Create**.
2. On the **Intelligent Structure Model** page, enter a name for the intelligent structure model.

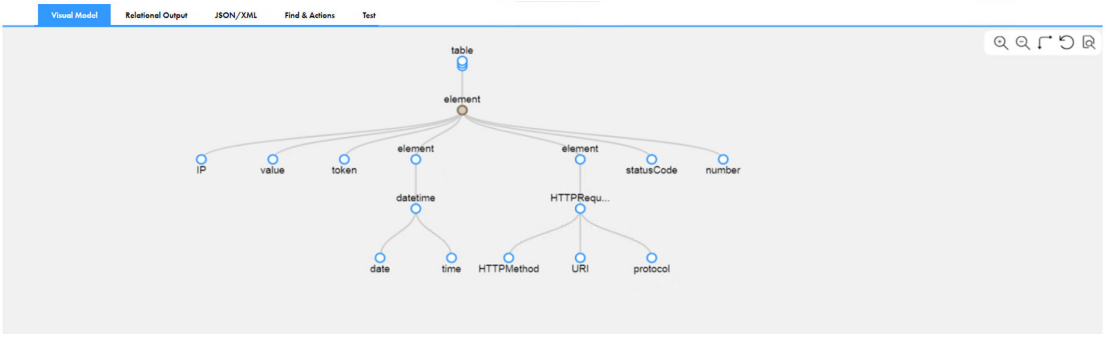
   The name can contain alphanumeric characters and underscores.
3. Navigate to the project and folder where you want to save the model, or use the default location.

   You can change the name or location after you save the intelligent structure model using the **Explore** page.
4. Based on the type of input you're using, perform one of the following steps:
   - To use a JSON sample file, first choose whether to base the model on a file sampling or on the entire file. Select the file and then click **Discover Structure**.
   - To use an XML sample file, first choose whether to base the model on a file sampling or on the entire file. Select the file, choose how you want to define the output groups, and then click **Discover Structure**.
   - To use an Avro schema file or any other type of sample file, select the file and click **Discover Structure**.

- To use an XSD schema file, first choose whether to base the model on a file sampling or, if the schema is larger than 1.5 MB, to base the model on the entire schema. Select the file, verify that the schema root is selected, choose how you want to define the output groups, and click **Discover Structure**. If you intend to use the model in a Structure Parser transformation, you can click **Upload XML Sample** and select an XML sample file to attach to the model.
- To use a Cobol copybook, select the copybook. If required, modify the values of **File organization** and **Code Page** to use at run time. Click **Discover Structure**.

For more information about input types, see "Inputs for intelligent structure models" on page 54.

After you click **Discover Structure**, Intelligent Structure Discovery deciphers the data in the input and discovers the patterns expressed in the data. The following image shows an example of discovered structure on the **Visual Model** tab:



Intelligent Structure Discovery creates nodes with unique names. If Intelligent Structure Discovery discovers instances of the same type of data, it adds a number suffix to the node names. For example, if the input contains timestamps in two tables, Intelligent Structure Discovery names them **timestamp1** and **timestamp2**.

5. When you base the model on an Avro, ORC, or Parquet file, Intelligent Structure Discovery discovers both the data elements and the elements of the file schema. By default, Intelligent Structure Discovery excludes elements that appear only in the schema from the model. To add schema elements to the output, include them in the structure of the model. For more information, see "Performing actions on multiple nodes" on page 76.
6. For a model that you create for an Excel worksheet, Intelligent Structure Discovery creates metadata nodes with sheet index and name. By default, Intelligent Structure Discovery excludes those nodes from the structure of the model. To add the nodes to the output, include them in the structure. For more information, see "Edit the structure of Microsoft Excel input" on page 78.
7. You can refine the structure so that when you use the model in production the output meets your requirements. For more information, see Chapter 10, "Refining intelligent structure models" on page 67.
8. Click **Save**.

   Intelligent Structure Discovery generates an intelligent structure model and saves the model in the selected location.

Exporting an intelligent structure model

Before you can use an intelligent structure model in a data engineering mapping, you must export it. You export the model to your local drive, and can then incorporate the model into data engineering data objects.

To export an intelligent structure model, your organization must have the appropriate license.

1. On the **Explore** page, navigate to the project and folder with the intelligent structure model.

   The **Explore** page shows all the assets in the folder.

2. Click to select the row that contains the relevant intelligent structure model. In the **Actions** menu, select **Edit**. In the **Intelligent Structure Details** panel, click **Edit**.

   The intelligent structure model appears in a separate page.

3. In the **Intelligent Structure** page, find the icon menu in the upper right corner of the Visual Model tab, and click the **Export model** icon.

   Note: The **Export model** icon is available only if your organization has the appropriate license.

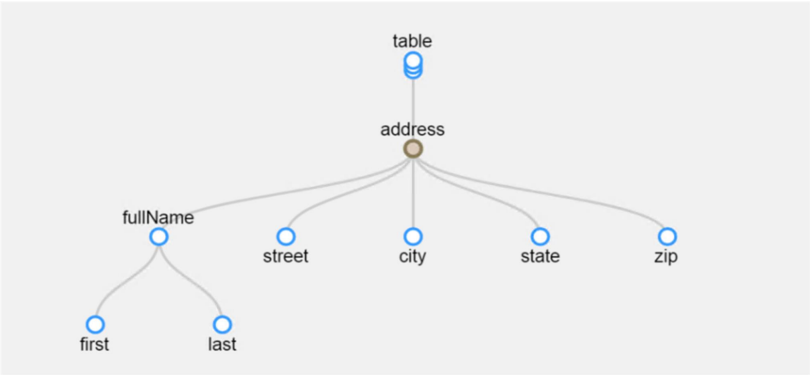
Intelligent structure model example

Intelligent Structure Discovery creates an intelligent structure model based on the structure of the input data that you provide.

As an example, you want to create a model for a CSV input file that contains the following content:

```
first,last,street,city,state,zip
Carrine,Stone,17 Torrence Street,Livingston,PA,10173
Poona,Tillkup,52 Perez Avenue,Livingston,PA,10256
Tasha,Herrera,158 Shiraz Boulevard,Kensington,WA,33823
John,Washington,22A Zangville Drive,Tucson,AZ,20198
Jane Hochuli 4483 Central Street Suite 30 Phoenix PA 38721
```

The following image shows the structure that Intelligent Structure Discovery discovers based on the input file:



You can see that Intelligent Structure Discovery created nodes representing the fields in the input file, such as **first**, **last**, **street**, **city**, **state**, and **zip**.

The structure represents not just the data fields themselves, but also the relationship of the fields to each other. For example, Intelligent Structure Discovery recognized that the data `Carrine,Stone` represents the first name and last name of a person. The nodes **first** and **last** are grouped together under the node **fullName**, representing the relationship of the data with each other.

Intelligent Structure Discovery also recognized that the data as a whole represented addresses. The data is grouped under a parent node **address**.

The nodes represent fields that are part of the output. Nodes that are related are grouped into an output group. Output groups can contain one or more nodes.

Use case

You work in an operations group for a streaming media service company. Your team wants to process web logs from your server farms to obtain operations analytics and to identify maintenance issues.

Your back-end system collects data regarding server access and system load in your server farms. Your team wants to identify the operations that have created the most server load in the past few weeks. You want to store data afterwards for auditing purposes.

Before your data analysts can begin working with the data, you need to parse the data. However, the logs are semi-structured, and after server upgrades the log file structure might change slightly and some of the information might take a different format. With a standard transformation, this would cause data loss or log processing failures.

If the input data contains headers, Intelligent Structure Discovery supports data drift to different locations. If the input data does not contain headers, Intelligent Structure Discovery identifies additional data at the end of the input.

Your initial log files have the following structure:

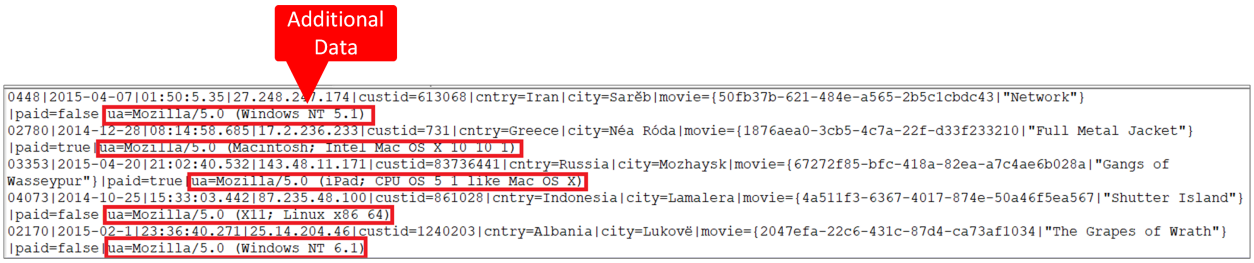
```
05967|2014-09-19|04:49:50.476|51.88.6.206|custid=83834785|cntry=Tanzania|city=Mtwango|
movie={b1027374-6eec-4568-8af6-6c037d828c66|"Touch of Evil"}|paid=true
01357|2014-11-13|18:07:57.441|88.2.218.236|custid=41834772|movie={01924cd3-87f4-4492-
b26c-268342e87eaf|"The Good, the Bad and the Ugly"}|paid=true
00873|2014-06-14|09:16:14.522|134.254.152.84|custid=58770178|movie={cd381236-53bd-4119-
b2ce-315dae932782|"Donnie Darko"}|paid=true
02112|2015-01-29|20:40:37.210|105.107.203.34|custid=49774177|cntry=Colombia|city=Palmito|
movie={ba1c48ed-d9ac-4bcb-be5d-cf3afbb61f04|"Lagaan: Once Upon a Time in India"}|
paid=false
00408|2014-06-24|03:44:33.612|172.149.175.30|custid=29613035|cntry=Iran|city=Bastak|
movie={3d022c51-f87f-487a-bc7f-1b9e5d138791|"The Shining"}|paid=false
03568|2015-01-07|11:36:50.52|82.81.202.22|custid=27515249|cntry=Philippines|
city=Magallanes|movie={ad3ae2b4-496e-4f79-a6dd-202ec932e0ae|"Inglourious Basterds"}|
paid=true
```

After server upgrades, some log files have the following structure:

```
0448|2015-04-07|01:50:5.35|27.248.247.174|custid=613068|cntry=Iran|city=SarÄ•b|
movie={50fb37b-621-484e-a565-2b5c1cbdc43|"Network"}|paid=false|ua=Mozilla/5.0 (Windows
NT 5.1)
02780|2014-12-28|08:14:58.685|17.2.236.233|custid=731|cntry=Greece|city=NÃ©a RÃ³da|
movie={1876aea0-3cb5-4c7a-22f-d33f233210|"Full Metal Jacket"}|paid=true|ua=Mozilla/5.0
(Macintosh; Intel Mac OS X 10_10_1)
03353|2015-04-20|21:02:40.532|143.48.11.171|custid=83736441|cntry=Russia|city=Mozhaysk|
movie={67272f85-bfc-418a-82ea-a7c4ae6b028a|"Gangs of Wasseypur"}|paid=true|
ua=Mozilla/5.0 (iPad; CPU OS 5_1 like Mac OS X)
04073|2014-10-25|15:33:03.442|87.235.48.100|custid=861028|cntry=Indonesia|city=Lamalera|
movie={4a511f3-6367-4017-874e-50a46f5ea567|"Shutter Island"}|paid=false|ua=Mozilla/5.0
(X11; Linux x86_64)
02170|2015-02-1|23:36:40.271|25.14.204.46|custid=1240203|cntry=Albania|city=LukovÃ«|
movie={2047efa-22c6-431c-87d4-ca73af1034|"The Grapes of Wrath"}|paid=false|
ua=Mozilla/5.0 (Windows NT 6.1)
```

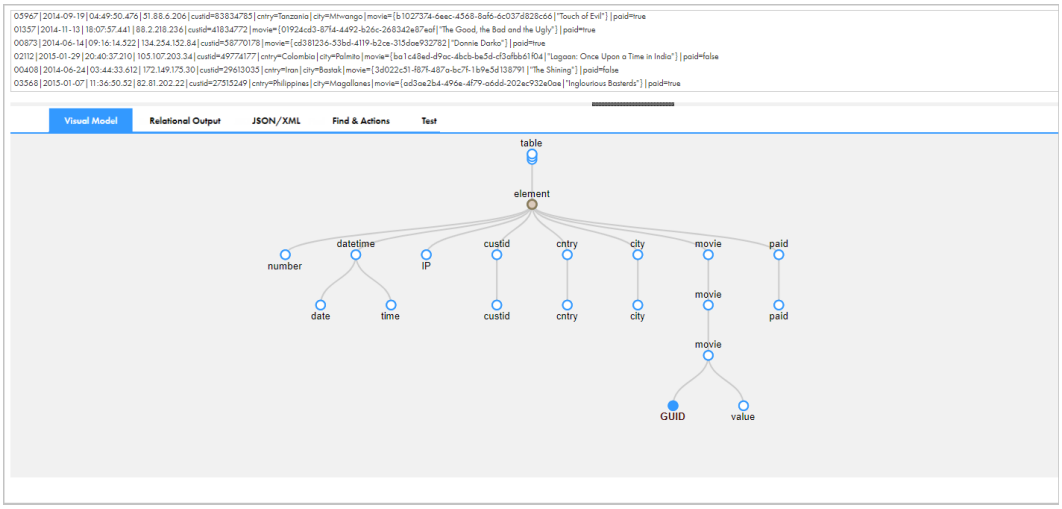
The data format varies, and some of the data has drifted to a different location.

The following image shows the data variations:



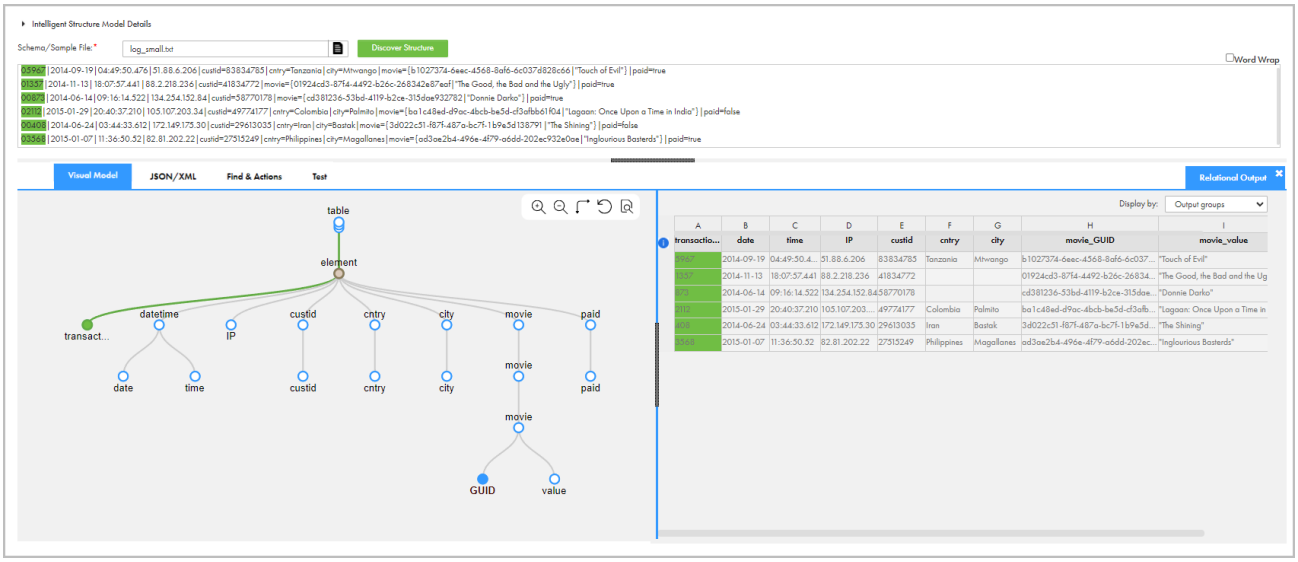
Instead of manually creating individual transformations, your team can create an intelligent structure model to determine the relevant data sets. You create an intelligent structure in Intelligent Structure Discovery and automatically identify the structure of the data.

The following image shows the intelligent structure that you create:



When you examine the data, you realize that the first element in the model, `number`, actually represents the user transaction identification. You change the element name to `transactionId`.

The following image shows the updated intelligent structure:



After you save the intelligent structure as an intelligent structure model, you create a Structure Parser transformation and assign the model to it. You can add the transformation to a Data Integration mapping with a source, target, and other transformations. After the mapping fetches data from a source connection, such as Amazon S3 input buckets, the Structure Parser processes the data with an intelligent structure model. The transformation passes the web log data to downstream transformations for further processing, and then to a target, such as Amazon S3 output buckets.

Troubleshooting intelligent structure models

Consider the following troubleshooting tips when you create intelligent structure models.

Using differently structured files causes data loss.

If the intelligent structure model does not match the input file or only partially matches the input file, there might be data loss.

For example, you created a model for a sample file that contains rows with six fields of data, `computer ID`, `computer IP address`, `access URL`, `username`, `password`, and `access timestamp`. However, some of the input files contained rows with eight fields of data, that is a `computer ID`, `computer name`, `computer IP address`, `country of origin`, `access URL`, `username`, `password`, `access code`, and `access timestamp`. The data might be misidentified and some data might be designated as unidentified data.

If some input files contain more types of data than other input files, or different types of data, for best results create a sample file that contains all the different types of data.

Data in a Microsoft Word or Microsoft Excel file wasn't parsed.

When Intelligent Structure Discovery creates a model that is based on a Microsoft Word or Microsoft Excel file, it might discover unstructured data as an unparsed node and exclude the node from the model structure and from the output, for example, when the file contains free text. You can edit the model to include excluded nodes in the structure. For more information, see "Editing the nodes" on page 72.

Data in PDF forms wasn't modeled or parsed.

An intelligent structure model parses data within PDF form fields. Ensure that the PDF form includes fields.

Error: Unsupported field names might cause data loss.

Do not use duplicate names for different elements.

If you use Big Data Management 10.2.1, ensure that the names of output groups follow Informatica Developer naming conventions. An element name must contain only English letters (A- Z, a-z), numerals (0-9), and underscores. Do not use reserved logical terms, and do not start element names with a number.

In later versions of Big Data Management or Data Engineering Integration, Intelligent Structure Discovery replaces special characters in element names with underscores and inserts underscores before element names that start with numerals and before element names that are reserved logical terms.

When you try to base a model on a sample ORC file that contains Union data, the model creation fails.

Intelligent Structure Discovery doesn't process the Union data type in ORC input. Select a file that doesn't contain Union data to base the model on.

CHAPTER 10

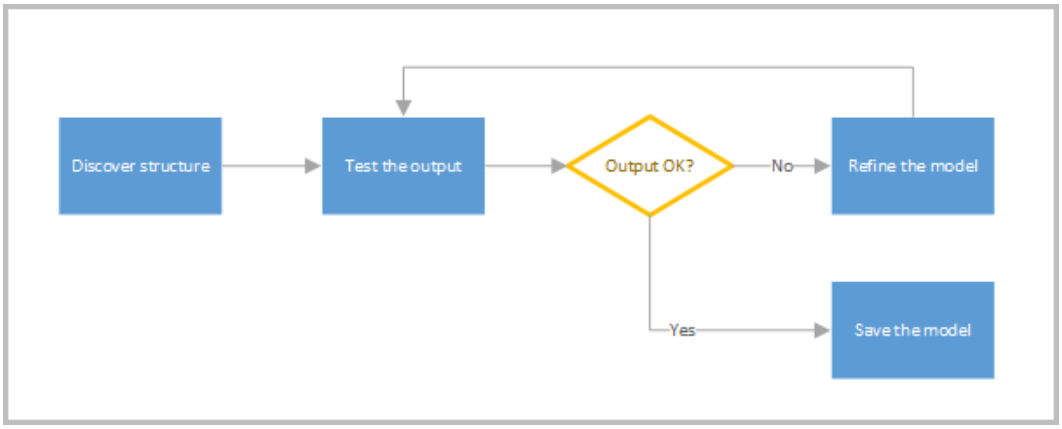
Refining intelligent structure models

After Intelligent Structure Discovery discovers the structure of the model input, you can refine the structure so that the output meets your requirements when you use the model in production.

Use the **Intelligent Structure Model** page to refine the model. For example, you can work with the visual model to understand the structure, find and rename a node, add a prefix to file names, or define rows or columns in Microsoft Excel files as table headers.

As you refine the discovered structure, you can test the output that Intelligent Structure Discovery generates based on the model in different output formats. Repeat refining and testing the model until you are satisfied that it generates the required output.

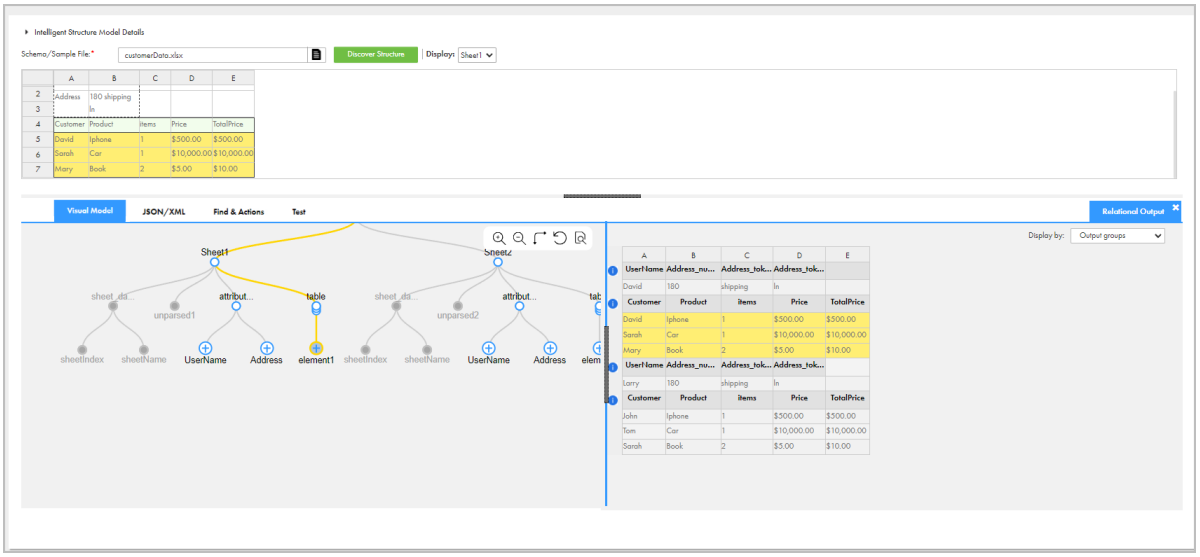
The following image shows the process of discovering, refining, testing, and saving a model:



Intelligent structure model views

The **Intelligent Structure Model** page includes different views to help you visualize the model, refine the structure, test the output, and configure settings.

The following image shows a sample of the **Intelligent Structure Model** page:



After you upload an input file and click **Discover Structure**, a panel shows a preview of the data file. When the sample file is an Excel file with multiple sheets, you can select which sheet to preview. When the input contains lengthy rows of data, you can select **Word wrap** to wrap each row. If you don't wrap lengthy input rows, only part of each row shows on the panel. Wrapping the text does not affect the intelligent structure or the input format. The panel is not available when the input is an XSD schema

Use the tabs on the **Intelligent Structure Model** page to refine the discovered structure. The following table describes the tabs:

Tab	Description
Visual Model	Review the discovered structure in the tree view or folder view. The visual model shows the discovered data types as nodes, and you can perform actions on the nodes, such as changing the root node, renaming a node, and splitting a node.
Relational Output	Edit the output groups that the intelligent structure generates.
JSON/XML	View the discovered structure in JSON or XML format.
Find & Actions	Search for structure elements, such as nodes and fields, and perform actions on multiple nodes.
Test	Test the output that Intelligent Structure Discovery generates based on the model.
Settings	Configure settings for XML and XSD input files. Before you discover the structure of a file, you can configure how Intelligent Structure Discovery defines the output groups. For more information, see "Output group definition" on page 56. After you discover the structure of the file, you can configure how the intelligent structure model handles leading and lagging spaces. By default, the model trims all leading and lagging spaces.

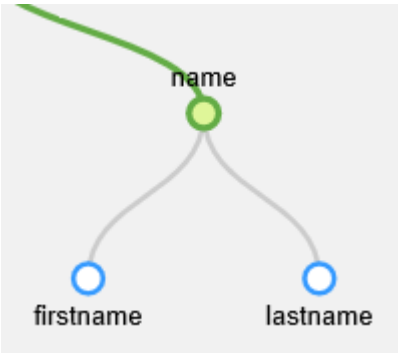
Working with the visual model

Use the **Visual Model** tab to view the structure of the intelligent structure model. The visual model shows the discovered types of data as nodes and shows their relationship to each other in a graphical format.

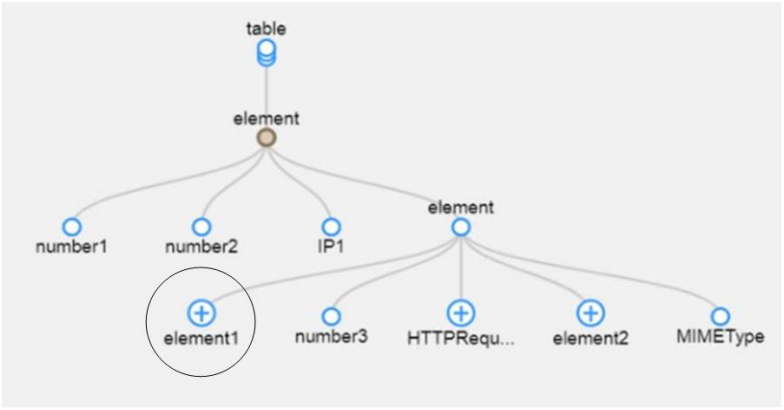
You can drag the model to focus on a specific section. You can also use the following actions to navigate the structure:

- Export. Exports the model. Available with the appropriate license.
- Zoom in and zoom out. Increases and decreases the display size of the model.
- Rotate. Toggles the display between the folder view and tree view.
- Undo. Reverts the last action.
- Find. Opens the **Find & Actions** tab.

In the tree view, a grouping of data fields appears as a node with child nodes. For example, the input data might contains a full name, such as `Tatanya Morales`. The intelligent structure model represents the name with a `name` parent node, a child node for the first name, and a child node for the last name. The following image shows the parent and child nodes:



You can collapse parent nodes, for example, if an intelligent structure model contains parent nodes with many child nodes. To collapse a node, click the node and select **Collapse**. The visual model hides the child nodes, but they remain in the intelligent structure model. The following image shows a model with a collapsed node:

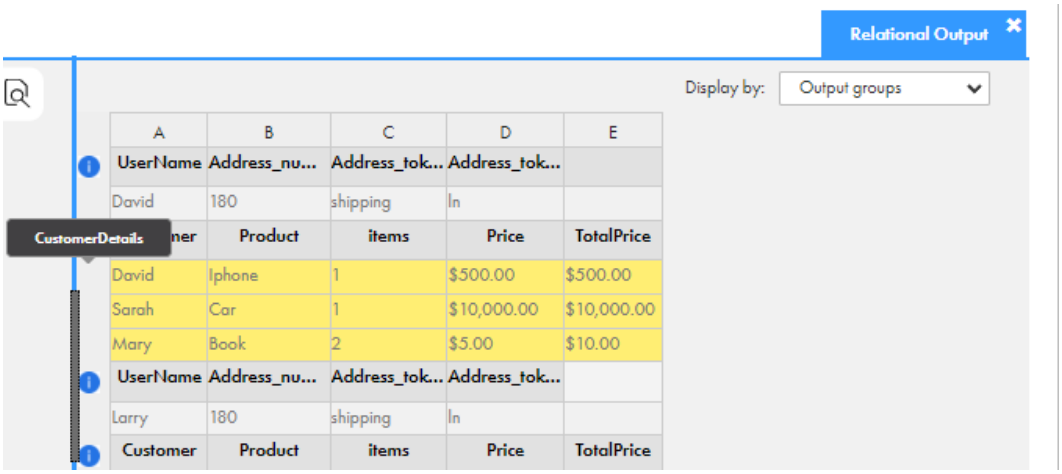


Note: If a model is based on an Excel file that contains multiple sheets, the visual model displays all of the sheets.

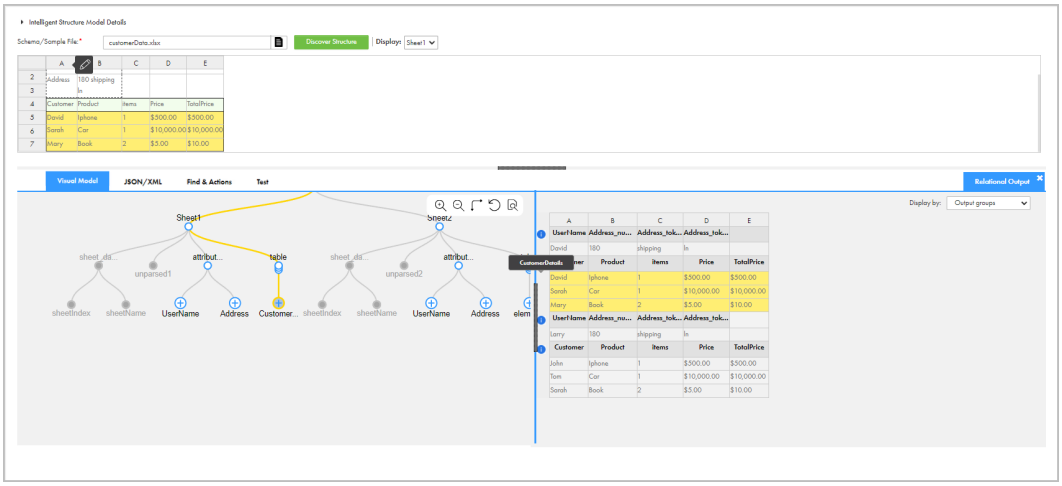
Viewing the relational output

The **Relational Output** tab shows the output that the intelligent structure model produces.

The output is organized in one or more output groups. An output group contains one or more nodes from the intelligent structure model. Each output group has an information icon to the left of its column display. When you hover over the information icon, the entire output group is highlighted and the name of the output group is displayed. The following image shows a highlighted outgroup group and the group name:



To better understand the output, you can select a node in the visual model and view the corresponding data in the input data panel and in the related column in the output group. As you refine the intelligent structure model on the **Visual Model** tab, you see the changes in the output groups on the **Relational Output** tab. The following image shows a highlighted output group, with the same data highlighted in the input data panel and the visual model:



You can select one of the following display options in the **Display by** field of the **Relational Output** tab:

- Display output groups. View the way data is grouped together in the output.
- Display incoming records. Understand how Intelligent Structure Discovery receives input records.

The display option you select doesn't affect the structure of the data when you run a mapping that uses the model.

Note: The **Relational Output** tab can display up to 200 rows of data. If you have a large amount of data or many repeating groups, the **Relational Output** tab might not display all the output groups.

Working with relational output

Use the **Relational Output** tab to perform actions on the output groups that affect the structure of the data when you run a mapping that uses the model.

You can perform the following actions on the **Relational Output** tab:

Configure data normalization

Use the **Data Normalization** field to choose whether the output groups are normalized or denormalized.

By default, when a model that is based on a JSON, XML, or XSD file contains nested repeating groups, Intelligent Structure Discovery normalizes the input data and assigns each nested repeating group to its output group. For models that are based on other input types, Intelligent Structure Discovery doesn't normalize the data by default.

Rename a node

To rename a node, right-click the column in the output group that represents the node and type the new name.

Exclude a node

To exclude a node from an output group, right-click the column and select **Exclude from Structure**. If you exclude the node, the data is not parsed when you run a mapping that uses the model.

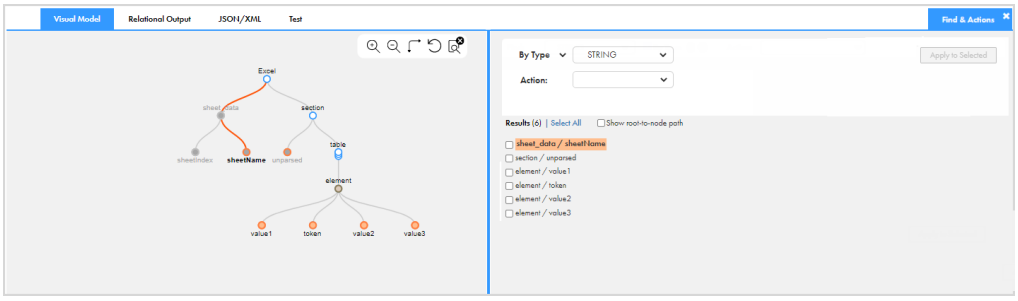
Finding nodes in a structure

Use the **Find & Actions** tab to locate nodes in a structure. You can search for nodes by name or by data type.

1. On the **Intelligent Structure Model** page, click the **Find & Actions** tab.
2. Perform one of the following actions to search for nodes:
   - To search for nodes by name, in the **Find** field, enter a string to search for and click the arrow.
   - To search for nodes by data type, select **By Type** and select the data type to search for. The list is divided into basic types and semantic data types and contains all the data types that are used in the model.

   The search results appear in the **Results** area and the nodes that you found are highlighted in the model on the **Visual Model** tab. For each node, the **Results** area shows the last two nodes of the path to the node.

The following image shows the results of a search by a STRING data type:



3. Optionally, to show the full path to the nodes, select **Show root-to-node path**.

Viewing and performing actions on node data

You can view information about a node in a structure, including the accuracy of the algorithm results, possible node types related to the node, and sample data in the input file that corresponds to the node.

You can perform actions on the data in the node. The actions you can perform depend on the type of sample data that you base the model on and on the output data type of the node.

1. On the **Intelligent Structure Model** page, on the **Visual Model** tab, double-click the node or right-click the node and select **Open Data**.

   The **Data for "<node name>"** dialog box appears.
2. To change the data type of the node, select the data type from the **Element Type** list. The data type selection applies to an intelligent structure model that you export to Informatica Developer.
3. Based on the data type of the node, perform one or more of the following actions:
   - To change the type format, enter the new format in the **Format** field.
   - To change the precision node, enter the value in the **Precision** field.

     Note: When you change the precision of an element in an existing intelligent structure model, the order of the output group elements in the Structure Parser transformation remains the same.
   - To change the scale, enter the value in the **Scale** field.
4. Click **OK**.

Editing the nodes

You can change the output of a model by editing the intelligent structure nodes. The available options depend on the type of sample input that you base the model on and on the output data type of the node.

You perform actions on nodes on the **Visual Model** tab.

Perform the following actions to edit the nodes:

Change the root node.

To change the root, right-click the new root node and select **Mark As Root**. Changes to root nodes don't affect the XML, JSON, or Relational outputs of the Structure Parser transformation.

Rename a node.

To rename a node, right-click a node, select **Rename**, and then enter the name, for example **Street Address**. Node names are not case sensitive. Names must be unique across sibling nodes, that is, nodes that reside under the same node.

Apply unique names to nodes

To apply unique naming across the structure, right-click the root node and select **Apply Unique Naming**. When you apply unique naming to a model, if the data in the sample input contains fields with identical names in different groups,Intelligent Structure Discovery uses suffixes to rename identical names to unique names.

Restore original node names.

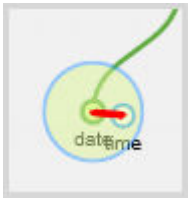
You can restore the original node names and remove the suffixes that Intelligent Structure Discovery adds to identical names in different hierarchy levels.

To restore the original node names in a structure, right-click the root node of the structure and select **Restore Original Names**.

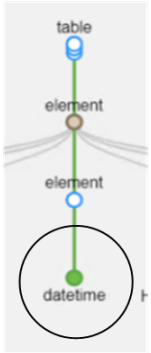
Note: Don't restore original node names for models that you plan to use in a Data Integration mapping with a relational output. Under these conditions, node names must be unique across the structure.

Combine data in two nodes.

To join two nodes, click and drag a node to the other node. For example, in the following model, the **time** node is selected and dragged to the **date** node:



The following image shows the combined node:



Combine all child node data into a parent node.

To flatten a parent node and merge its child nodes, right-click the node and select **Flatten**. When you flatten a node, the data from the child nodes is merged into one output field for the parent node. The child nodes are no longer separate fields in the output.

Exclude a node from the structure.

To exclude a node from the structure, right-click the node and select **Exclude from Structure**. When you exclude a node from the structure, the data from the node is not part of the output of the model and is not parsed during run time.

Include a node in the structure.

You can re-include excluded nodes in the structure and include nodes that Intelligent Structure Discovery doesn't include in the structure by default. For example, you can include the name of an Excel spreadsheet in the structure. To include a node in the structure, right-click the node and select **Include in Structure**.

Determine how Intelligent Structure Discovery parses data from Microsoft Excel files.

You can determine whether Intelligent Structure Discovery parses the formatted data that Microsoft Excel shows or the underlying, unformatted data when it parses fields in a Microsoft Excel file.

To determine how Intelligent Structure Discovery parses the data, right-click the node and select either of the following options:

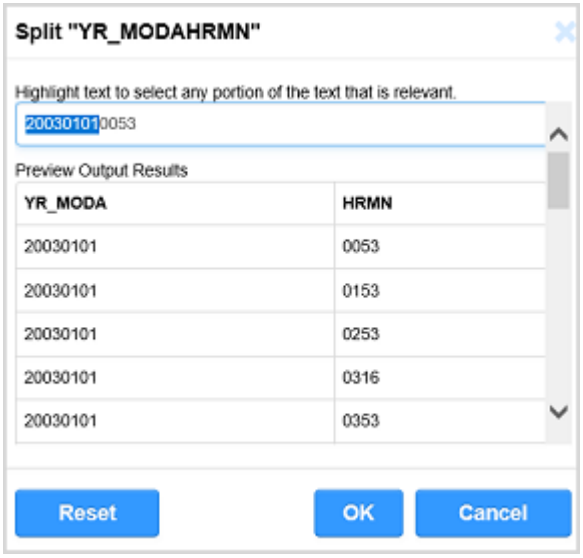
- Extract Formatted Data. Intelligent Structure Discovery parses the fields as formatted data.
- Extract Unformatted Data. Intelligent Structure Discovery parses the fields as unformatted data.

Split a node.

You can split nodes in a structure that is based on fixed-width files. To split the data in a node into two nodes, for example, for a credit card number for which you only want to store the last four digits, perform the following steps:

1. Right-click the node and select **Split**. The **Split** dialog box appears.
2. In the **Split** dialog box, highlight the data in the display field to indicate where to split the data. The data is split and displayed as two fields, with one field containing the highlighted data, and the second field containing the non-highlighted data.
3. To confirm the split, click **OK**.
4. To undo the split, click **Reset**.

For example, you want to split a node with year-month-date-hour-minute data in one field into two fields. You want to create a year-month-date field and an hour-minute field. The following image shows the highlighted section that you want to split:



Treat JSON model as repeating

By default, when the sample JSON model does not contain repetitions, the Structure Parser transformation does not parse repeating groups. If you want to parse repeating groups, right-click the JSON root node and select **Treat as Recurring**.

Treat an element as repeating

By default, when an element in the sample JSON or XML model does not contain repetitions, the Structure Parser transformation does not parse repetitions of the element. If you want to parse repetitions of the element, right-click the node and select **Treat as a List**. To undo a **Treat as a List** action, right-click the node and select **Treat as Single**.

Working with repeating groups

The output for groups is relational tables. When the intelligent structure contains repeating groups that relate to each other, you can configure the relational output to contain different tables for nested groups, as well as select a record ID.

When an intelligent structure contains nested repeating groups, you can perform the following actions on the **Visual Model** tab:

Promote to group.

To reduce the number of ports in the model, assign each nested repeating group to its own output group. When you promote a group that is nested within a repeating group, Intelligent Structure Discovery creates a record ID in the parent output group and a foreign key in the child output group. To promote a group to an independent output group, right-click the nested repeating node and select **Promote to Group**.

Note: For models that are based on a JSON, XML, or XSD file, Intelligent Structure Discovery assigns nested repeating groups to their own output groups when it creates the model.

Join to parent group.

Re-include a nested group in the parent group after you assigned it to its own output group. To re-include a group, right-click the nested repeating node and select **Join to Parent Group**.

Select as record ID.

Change the record ID. Select a node in the parent group and assign it as a record ID. Intelligent Structure Discovery automatically changes the foreign key in the nested group. To assign a node as record ID, right-click the node and select **Select as Record ID**.

Deselect as record ID.

Reinstate the initial record ID and foreign key for the repeating groups. To deselect a node as the record ID, right-click the node and select **Deselect as Record ID**.

Adding document identifiers to data in a model

When a model that is based on a JSON, XML, or XSD file contains non-repeating output groups, Intelligent Structure Discovery adds a document identifier to each non-repeating output group.

For models that are based on other input types, you can add document identifiers to output groups manually.

For example, add document identifiers to a model if you want the Structure Parser transformation to join output groups that are based on the same input and then output them to a single target. You can also use the document identifiers to join groups with the Joiner transformation.

To add a document identifier to an output group, right-click the node that represents the group and select **Add a Document ID**. To remove the identifier, right-click the node and select **Remove a Document ID**.

In the model, document identifier fields appear in the **Relational Output** tab, in the following format: `<group name>_documentId`. For example, for a group named "organization" the document identifier field is `organization_documentId`.

In the Structure Parser transformation, a document identifier output field appears for each output group.

Adding prefixes and suffixes to field names

You can add a prefix or a suffix to a field name or to multiple field names in a discovered structure.

Prefixes and suffixes can contain alphanumeric characters and underscores.

1. On the **Find & Actions** tab, search for the field name or for a part of the field name.

   Fields names that match the search appear on the **Find & Actions** tab.
2. Select the fields to add the prefix or suffix to, and then, from the **Actions** menu, select **Add Prefix** or **Add Suffix**.
3. Enter the prefix or suffix in the text field to the right of the **Actions** menu, and then click **Apply to Selected**.

   Intelligent Structure Discovery adds a prefix or a suffix to the names of the selected fields.

Performing actions on multiple nodes

You can perform actions on multiple nodes. The actions you can depend on the type of sample data that you base the model on and on the output data type of the node.

1. Search for nodes on the **Find & Actions** tab and select the nodes to perform the action on in the **Results** list.
2. From the **Action** list, select one of the following actions:

Option	Description
Add Prefix	Add a prefix to node names. Enter the prefix in the **With** field.
Add Suffix	Add a suffix to node names. Enter the suffix in the **With** field.
Exclude	Exclude the nodes from the intelligent structure. Data from the node is not part of the output of the model and is not parsed during run time.
Include	Re-include excluded nodes in the structure.
Expand	Expand the nodes to show portions of the model that are collapsed under the nodes.
Collapse	Hide portions of the model that are nested under the nodes.

Option	Description
Update Data Type	Change the data type of the nodes. Select the data type in the **To** field. Based on the data type that you select, you can edit the precision, scale, or type format of the nodes. The data type selection applies to an intelligent structure model that you export to Informatica Developer.
Add Child Prefix	Add a prefix to the child node names. Enter the prefix in the **With** field. **Note:** This command only applies prefixes to immediate child nodes. Prefixes don't apply to child nodes further down the node path.
Add Child Suffix	Add a suffix to the child node names. Enter the suffix in the **With** field. **Note:** This command only applies suffixes to immediate child nodes. Suffixes don't apply to child nodes further down the node path.
Change Type to String	Change the data type of the nodes to string.
Replace	Replace a string in node names with the string you enter in the **With** field. Node names are not case sensitive. Names must be unique across sibling nodes, that is, nodes that reside under the same node.
Discover data type by content	Discover the data type of numbers that are enclosed in double quotes based on the content of the node that contains the data. Applies to JSON files. **Note:** Once you apply this action to a node or to multiple nodes, you can't select to revert to the default mode of identifying numbers that are enclosed in double quotes as strings. To re-apply the default mode you must select the sample file and discover the structure again.
Extract Formatted Data	Parse the formatted data that Microsoft Excel shows. Applies to Microsoft Excel files.
Extract Unformatted Data	Parse the underlying, unformatted data. Applies to Microsoft Excel files.

3. Click **Apply to Selected**.

Enriching an existing model with new samples

After you create a model based on a JSON, XML, ORC, AVRO, or PARQUET sample file, you can use additional sample files to enrich the structure with fields that exist in the new samples.

For example, if the data you want to map contains data that is not in the structure and therefore is not mapped, you can add that data to the structure based on a new sample file that contains that data.

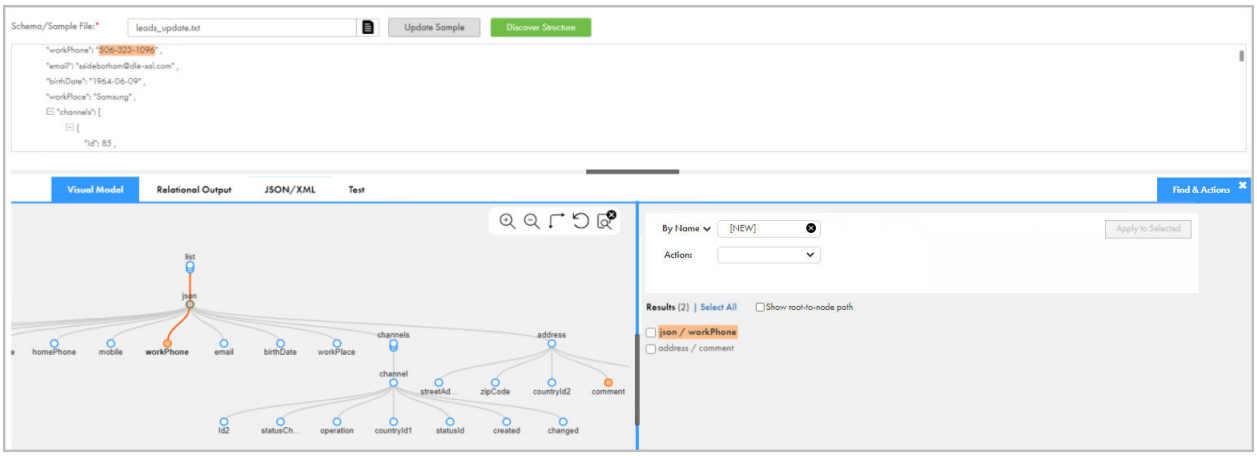
The files that you use to enrich the structure must be of the same file type as the type of file that the structure is based on. For example, to enrich a structure that is based on a JSON sample file use additional JSON sample files.

To add a new sample file to the structure click **Update Sample** and select the new sample file.

Intelligent Structure Discovery creates nodes for new data in the sample file. To facilitate locating the new nodes, the **Find & Actions** tab opens with the search pattern **[NEW]**. The list of new nodes shows and those nodes are highlighted in the structure.

Note: If the new sample does not contain all the data that was in the structure, no data is removed from the structure and the entire structure is displayed. However, the input data panel and the **Relational Output** tab only show data that is in the new sample.

The following image shows the updated model with data from the new sample:



When you save the model the structure is updated with the new data.

Edit the structure of Microsoft Excel input

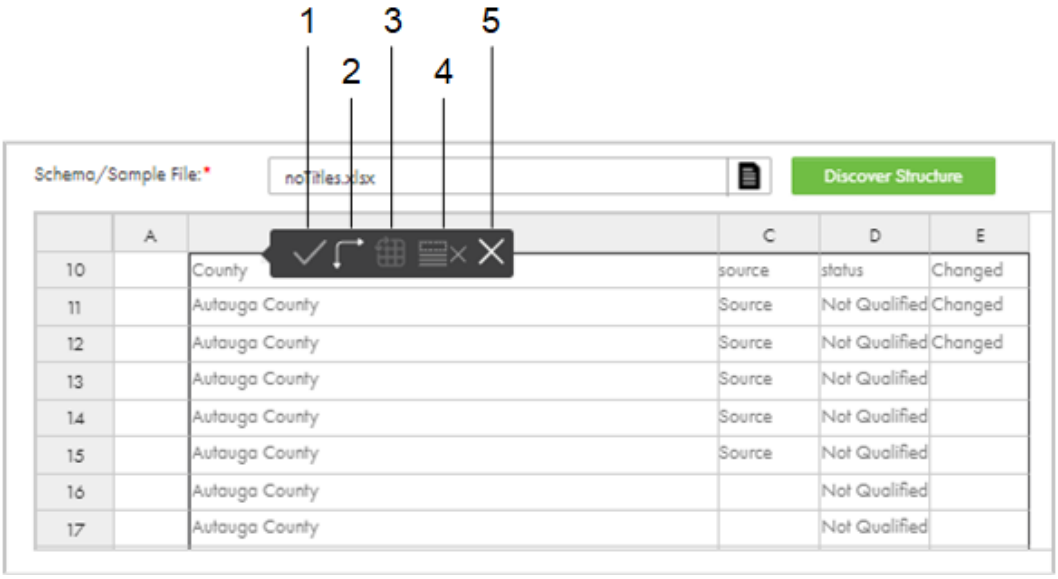
When the input of a model is a Microsoft Excel file, by default, Intelligent Structure Discovery doesn't include the metadata of the Excel spreadsheet in the structure, including the name and index number of the spreadsheet.

To include the spreadsheet name or index number in the structure, on the **Visual Model** tab of the **Intelligent Structure Model** page, right-click the sheetName or sheetIndex node, respectively, and select **Include in Structure**.

On the input data panel of the **Intelligent Structure Model** page, Intelligent Structure Discovery shows the model data in table format. When you hover over the table, the **Edit** button appears. Clicking the button opens a menu that you can use to perform the following actions on the table:

- Transpose the table, that is, change the orientation of table headers from rows to columns or from columns to rows.
- Convert a two-column or a two-row table to name-value pairs and convert name-value pairs to a table.
- Define table headers.

The following figure shows a sample input data panel for a Microsoft Excel input file and the menu that you use to edit the table:



1. Apply
2. Transpose
3. Convert to name-value pairs/Convert to table
4. Remove all headers
5. Cancel

Transposing a table

You can transpose tables in models that are based on Microsoft Excel files.

1. On the **Intelligent Structure Model** page, on the input data panel, hover over the table and click the **Edit** button.
2. Click the **Transpose** button.

   The orientation of the table changes. The change shows on the input data panel and in the model on the **Visual Model** tab.

Switching between a table and name-value pairs

You can switch between a two-column or two-row table and name-value pairs in models that are based on Microsoft Excel files.

1. On the **Intelligent Structure Model** page, on the input data panel, hover over the table and click the **Edit** button.
2. Click the **Convert to name-value pairs** or the **Convert to table** button, as applicable.

   Note: The **Convert to table** button is enabled only for two-column and two-row tables. You can't convert larger tables to name-value pairs.

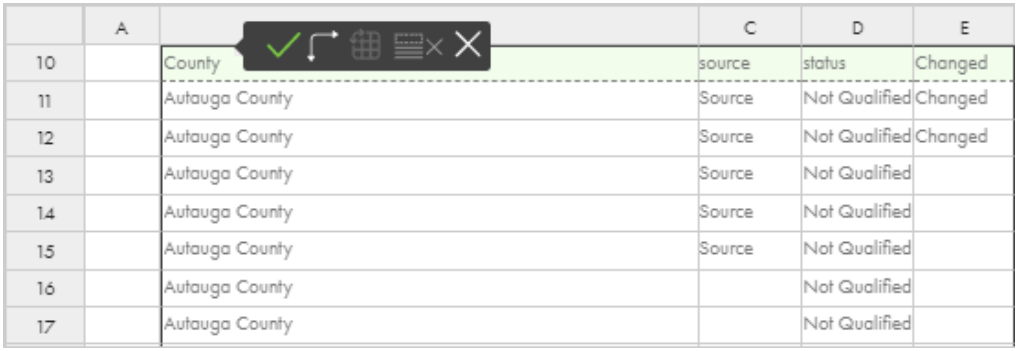
   The revised structure shows on the input data panel and in the model on the **Visual Model** tab.

Defining table headers

You can define rows and columns as table headers in models that are based on Microsoft Excel files.

1. On the **Intelligent Structure Model** page, on the input data panel, hover over the table, click the **Edit** button, and select the required rows or columns.

   The selected rows or columns appear in green.



	A		C	D	E
10		County	source	status	Changed
11		Autauga County	Source	Not Qualified	Changed
12		Autauga County	Source	Not Qualified	Changed
13		Autauga County	Source	Not Qualified	
14		Autauga County	Source	Not Qualified	
15		Autauga County	Source	Not Qualified	
16		Autauga County		Not Qualified	
17		Autauga County		Not Qualified	

2. Click the **Apply** button.

   The selected rows or columns show as headers in the model on the **Visual Model** tab.

3. To clear the selection of table headers, on the input data panel, click the **Remove all headers** button.

Testing the output

After Intelligent Structure Discovery discovers the structure of the input that you provide for the model, you can test the model output. You can test different output formats for the model.

1. On the **Intelligent Structure Model** page, select the **Test** tab.
2. Select the output format to test from the **Test Output Format** list.
3. Click **Select and Transform**, select the input file to test, and click **Open**.

   Intelligent Structure Discovery saves the output to a .zip file in your default downloads folder.

4. Unzip the .zip file and open the output file to view the output that Intelligent Structure Discovery generated. If the input contains unidentified data, Intelligent Structure Discovery saves the unidentified data in a separate file.

CHAPTER 11

Mapplets

A mapplet is reusable transformation logic that you can use to transform source data before it is loaded into the target. If you are an Informatica PowerCenter user, you can import a PowerCenter mapplet to use in Data Integration.

You can create a mapplet in one of the following ways:

- Create a mapplet in Data Integration.
- Import a mapplet from PowerCenter. For information about importing a mapplet, see "PowerCenter mapplets" on page 87.
- Generate an SAP BAPI or IDoc mapplet. For information about configuring and importing SAP mapplets, see the help for SAP Connector.

Create mapplets in Data Integration in the Mapplet Designer in the same way that you create mappings in the Mapping Designer. After you create a mapplet, you can add it to a Mapplet transformation to use its transformation logic. You can use any transformation in a mapplet except the Web Services transformation.

You can use a mapplet in another mapplet. However, you cannot cyclically reference a mapplet. For example, if Mapplet A uses Mapplet B, Mapplet B cannot also use Mapplet A.

You can't use a mapplet in a mapping in SQL ELT mode.

Active and passive mapplets

Mapplets can be either active or passive.

Passive mapplets contain a single input group, a single output group, and only passive transformations.

Active mapplets contain at least one active transformation.

When you use a mapplet in a Mapplet transformation, the mapplet type displays on the **Mapplet** tab.

Mapplet input and output

To use a mapplet in a Mapplet transformation, you must configure the mapplet input and output.

A mapplet receives input from an upstream transformation through an Input transformation, from a Source transformation, or both.

A mapplet passes output to a downstream transformation through an Output transformation, to a Target transformation, or both.

A mapplet must contain at least one Input transformation or one Output transformation.

Mapplet input

Mapplet input can be an Input transformation, a Source transformation, or both.

Use an Input transformation when you want the mapplet to receive input data from one or more upstream transformations. You can use multiple Input transformations in a mapplet. When you use the mapplet in a Mapplet transformation, each Input transformation becomes an input group. Use multiple Input transformations when you have multiple pipelines in a mapplet, or when you want the mapplet to receive input from multiple upstream transformations.

You include one or more Source transformations in a mapplet to provide source data. When you use only Source transformations for mapplet input, the mapplet is the first object in the mapping pipeline and contains no input groups. If the Source transformation reads hierarchical data, you must specify the validation type in the Source transformation.

A mapplet must contain at least one Input transformation or Source transformation.

Note: Not all source types can be used in a Source transformation in a mapplet. For more information, see the help for the appropriate connector.

For information about configuring Input and Source transformations, see *Transformations*.

Mapplet output

Mapplet output can be to an Output transformation, a Target transformation, or both.

Use an Output transformation when you want the mapplet to pass data to one or more downstream transformations. When you use the mapplet in a Mapplet transformation, each Output transformation becomes an output group. Each output group can pass data to one or more pipelines in a mapping.

Use a Target transformation when you want the mapplet to write data to a target. When you use a Target transformation with no Output transformation, the mapplet is the last object in the mapping pipeline.

A mapplet must contain at least one Output transformation or Target transformation.

For information about configuring Output and Target transformations, see *Transformations*.

Transformation names

When you use a mapplet in a Mapplet transformation, Data Integration renames the transformations in the mapplet at run time.

When you use a mapplet in a Mapplet transformation, the mapplet might contain transformations with names that conflict with other transformation names in the mapping. To avoid name conflicts, Data Integration prefixes the names of the transformations in the mapplet with the Mapplet transformation name.

For example, a mapplet contains an Expression transformation named Expression_1. You create a mapping and use the mapplet in the Mapplet transformation Mapplet_Tx_1. When you run the mapping, the Expression transformation is renamed to Mapplet_Tx_1_Expression_1.

If a mapplet contains another Mapplet transformation, Data Integration also prefixes the transformation names with the second Mapplet transformation name. For example, the mapplet in the previous example also contains a Mapplet transformation named MappletTx, which contains FilterTx_1. In the mapping, FilterTx_1 is renamed to Mapplet_Tx_1_MappletTx_FilterTx_1.

Data Integration truncates transformation names that contain more than 80 characters.

Note: Data Integration only renames transformations in mapplets when the mapplet is used in a mapping created after the April 2022 release.

Parameters in mapplets

You can use input parameters and in-out parameters in a mapplet. You specify the value of the parameters when you configure the mapping task.

You configure parameters in mapplets the same way that you configure parameters in mappings. For information about configuring parameters, see *Mappings*.

When you include parameters in a mapplet, Data Integration renames the parameters when you use the mapplet in a Mapplet transformation. The parameter names are prefixed with the name of the Mapplet transformation. You can view the corresponding names on the **Parameters** tab of the Mapplet transformation Properties panel.

For example, you use a Filter transformation in a mapplet and select **Completely Parameterized** as the filter condition. You create a string parameter called "Filter_Param." You use the mapplet in a Mapplet transformation named "MyMappletTx." In the Mapplet transformation, the filter parameter is renamed to "MyMappletTx_Filter_Param."

If you define in-out parameters in a mapplet, the mapplet is not valid for mappings in advanced mode, even when the in-out parameters are not used in the mapplet.

Creating a mapplet

Create a mapplet in the Mapplet Designer.

The Mapplet Designer contains the same design areas and functionality as the Mapping Designer. For information about using the Mapping Designer, see *Mappings*.

1. Click **New** > **Mapplets** > **Mapplet**.

   The Mapplet Designer appears with an Input transformation and an Output transformation on the canvas.

2. Specify the mapplet name and location.

   The default name for the mapplet is `Mapplet` followed by a sequential number.
   If the **Explore** page is currently active and a project or folder is selected, the default location for the asset is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.

   You can change the name or location after you save the mapping using the **Explore** page.

3. Optionally, enter a description for the mapplet.

   When you use the mapplet in a Mapplet transformation, the description appears in the **Mapplet** tab.

4. Optionally, filter the transformations in the transformation palette.
5. Configure the mapplet input.

   Input can be a Source transformation, an Input transformation, or both. If you use an Input transformation, add input fields for the fields you want to pass to the mapplet.
6. Add transformations to the mapplet.
7. Configure the mapplet output.

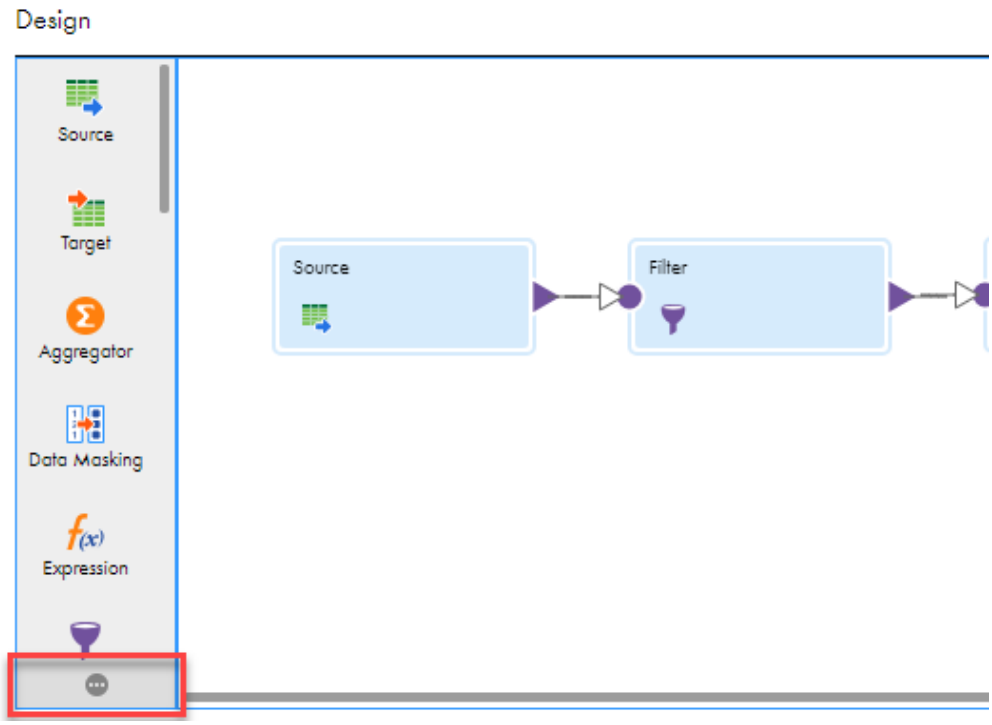
   Output can be a Target transformation, an Output transformation, or both. If you use an Output transformation, add output fields for the data you want to pass to the downstream transformation.
8. Validate the mapplet.
9. Resolve errors on the appropriate tab in the **Validation** panel.
   - Resolve errors on the **Mapping** tab to use the mapplet outside of advanced mode.
   - Resolve errors on the **Mapping (Advanced Mode)** tab to use the mapplet in advanced mode.
10. Save the mapplet.

Filtering the transformation palette

View licensed transformations, as well as transformations that are available in and outside of advanced mode, by filtering the transformation palette in the Mapplet Designer.

1. In the Mapplet Designer, click the **More** icon in the transformation palette.

   The following image shows the icon:



2. In the **Filter the Transformation Palette** dialog box, select one of the following mapping options:

- Mapping. View transformations that are available outside of advanced mode.
- Mapping (Advanced Mode). View transformations that are available in advanced mode.
- Shared. View transformations that are available both in and outside of advanced mode.
- All. View all transformations.

3. Select one of the following licensing options:
   - Licensed. View transformations available for your organization.
   - All. View all transformations.

Editing a mapplet

Edit a mapplet in the Mapplet Designer. To edit a mapplet, open the mapplet from the Explore page.

When you edit a mapplet, the Mapplet Designer validates the transformation logic. If a mapplet is valid for mappings in or outside of advanced mode when you save it, Data Integration deploys the changes to each mapping and mapplet that uses the mapplet.

For example, if the mapplet is valid both in and outside of advanced mode and you change the interface of the mapplet so that it's only valid outside of advanced mode, Data Integration deploys the changes to all mappings that use the mapplet.

Use caution when you edit mapplets that are used in mappings and other mapplets. If you edit the mapplet interface, mappings and mapplets that use the mapplet will have validation errors. To resolve the validation errors, you must synchronize the Mapplet transformations that use the mapplet to get the latest changes. If you do not synchronize the mapplet transformation and you run the task, the task fails.

To see the assets that use the mapplet, in the Mapplet Designer, open the **Actions** menu and select **Show Dependencies**.

You cannot edit a mapplet in a Mapplet transformation, and you cannot edit a mapplet that you imported from PowerCenter.

Changes that affect dependencies

Changes to a mapplet might affect the mappings and mapplets that use the mapplet.

The following changes to a mapplet do not affect the assets that use the mapplet:

- Change transformation names, descriptions, or properties.
- Add or remove transformations in the mapplet, as long as you do not change the mapplet type from active to passive or from passive to active.

The following changes to a mapplet change the interface of the mapplet and affect the assets that use the mapplet:

- Add or remove transformations that change the mapplet type from active to passive or from passive to active.
- Change the data type, precision, or scale of a connected input or output field.
- Add or remove input or output fields
- Add or remove Input or Output transformations

Synchronizing a mapplet

If the interface of a mapplet changes after it has been added to a Mapplet transformation, you must synchronize the mapplet to get the changes. Synchronize a mapplet on the **Mapplet** tab.

Mappings and mapplets that use the mapplet are invalid until the mapplet is synchronized. If you run a mapping task that includes a changed mapplet, the task fails.

When you synchronize a mapplet, the updates might cause validation errors in other transformations in the mapping or mapplet.

You cannot synchronize a mapplet that you imported into Data Integration from PowerCenter or SAP.

To synchronize a mapplet, perform the following steps:

1. Open the mapping or mapplet that uses the mapplet.
2. Select the mapplet transformation.
3. On the **Mapplet** tab, click **Synchronize**.
4. Correct any resulting errors in the transformation logic.

Data classifications

If your organization uses Metadata Command Center, you can link mapplets to data classifications that are associated with objects of a catalog source in Metadata Command Center. When you use the source object in a mapping, CLAIRE recommends the mapplets based on the data classifications associated with the object columns.

Before you can link data classifications to mapplets, in Metadata Command Center, create the data classifications, configure them with a catalog source, and execute a data classification scan job. In Data Integration, associate the data classifications with mapplets so that CLAIRE recommends the mapplets when you use sources that are also associated with the data classification.

You can add data classifications to valid mapplets that you create in Data Integration. You cannot add data classifications to a mapplet that you import from PowerCenter or SAP.

Adding data classifications

Add data classifications to a mapplet so that CLAIRE recommends the mapplet in mappings when you add a source object with a matching data classification.

1. On the **Explore** page, navigate to the mapplet.
2. In the row that contains the mapplet, click **Actions** > **Edit Data Classifications**.
3. In the **Data Classifications** window, drag and drop the data classifications that you want to associate with the mapplet.
4. Click **Save**.

Validating a mapplet

Validate a mapplet by resolving the errors in the expression editor and on the appropriate tab in the **Validation** panel.

Resolve the validation errors in the following areas of the Mapplet Designer:

Validation errors in the expression editor

The expression editor validates an expression for use both in and outside of advanced mode. The expression editor shows a different number of validation messages based on the processing differences between the Data Integration Server and an advanced cluster.

If the expression is valid or the validation error is the same for use both in and outside of advanced mode, the expression editor shows one validation message. If the expression is valid only in or outside of advanced mode, or the validation errors are different, the expression editor shows multiple validation messages.

Validation errors in the Validation panel

The **Validation** panel shows mapplet errors. To use the mapplet outside of advanced mode, resolve the errors on the **Mapping** tab. To use the mapplet in advanced mode, resolve the errors on the **Mapping (Advanced Mode)** tab.

If the mapplet is valid on one tab, the mapplet can be invalid on the other tab. If the mapplet contains no validation errors on either tab, you can use the mapplet in any mapping.

The following table lists the mapplet states and the mappings where you can use the mapplet:

Mapplet state	Mappings where you can use the mapplet
Valid	Can be used in all mappings
Invalid	Cannot be used in any mappings
Valid (For Mappings)	Can be used in mappings outside of advanced mode
Valid (For Mappings in Advanced Mode)	Can be used in mappings in advanced mode

PowerCenter mapplets

If you are an Informatica PowerCenter user, you can use a PowerCenter mapplet to create a mapplet in Data Integration.

To use a PowerCenter mapplet, you create a mapplet in PowerCenter and export the mapplet to an XML file. Then you upload the XML file in to Data Integration.

A mapplet contains a set of transformations. A PowerCenter mapplet can contain one or more Source transformations but it cannot contain a Target transformation. You can use PowerCenter mapplets in the following Data Integration tasks:

- Synchronization tasks. You can use one mapplet in a synchronization task.
- Mapping tasks. You can use multiple mapplets in a mapping task. You can use mapplets in a Visio template or in a mapping. You cannot use a PowerCenter mapplet in advanced mode.

- Masking tasks. You can use passive mapplets in a masking task to mask target fields.

Active and passive PowerCenter mapplets

Mapplets are either active or passive.

When you upload a PowerCenter mapplet in Data Integration, you specify whether the mapplet is active or passive.

An active mapplet includes at least one active PowerCenter transformation. An active mapplet can return a number of rows different from the number of source rows passed to the mapplet. For example, an active mapplet might aggregate five source rows and return one row.

A passive mapplet includes only passive PowerCenter transformations. A passive mapplet returns the same number of rows that are passed from the source.

Stored Procedures in mapplets

When a PowerCenter mapplet that you want to use contains a Stored Procedure transformation, the stored procedure must include exception handling.

Exception handling can be as complex as necessary. Or, you can use the following simple example:

```
Exception
when NO_DATA_FOUND
then NULL;
END;
```

For example, you have the following stored procedure in a PowerCenter workflow:

```
CREATE OR REPLACE PROCEDURE SP_GETSAL_WITH_EXCEPTION (EMP_ID NUMBER, EMP_NAME OUT
VARCHAR, SAL OUT NUMBER)
  AS
  BEGIN
  SELECT EMPNAME INTO EMP_NAME FROM EMPLOYEE WHERE EMPID=EMP_ID;
SELECT SALARY INTO SAL FROM EMPLOYEE WHERE EMPID=EMP_ID;
```

Before you export the workflow, add exception handling as follows:

```
CREATE OR REPLACE PROCEDURE SP_GETSAL_WITH_EXCEPTION (EMP_ID NUMBER, EMP_NAME OUT
VARCHAR, SAL OUT NUMBER)
  AS
  BEGIN
  SELECT EMPNAME INTO EMP_NAME FROM EMPLOYEE WHERE EMPID=EMP_ID;
SELECT SALARY INTO SAL FROM EMPLOYEE WHERE EMPID=EMP_ID;
Exception
when NO_DATA_FOUND
then NULL;
END;
```

PowerCenter XML files for mapplets

To use a mapplet in Data Integration, you upload a PowerCenter XML file that defines a PowerCenter mapplet.

Consider the following rules when you use a PowerCenter XML file to create a Data Integration mapplet:

- If the mapplet includes a transformation that uses a connection, then the PowerCenter XML file must contain only one workflow, one Session task, one mapping, and one mapplet.
  If the mapplet doesn't include a transformation that uses a connection, then the PowerCenter XML file must include one mapplet. The workflow, Session task, and mapping are optional.
- The session can use any type of connection.

- You do not have to map all source and target fields in the PowerCenter mapping.
- The PowerCenter mapplet can contain the following supported transformations:
  - Aggregator transformation
  - Expression transformation
  - Filter transformation
  - HTTP transformation
  - Lookup transformation
  - Salesforce Lookup transformation (multiple matches returns a single match)
  - Salesforce Picklist transformation
  - Salesforce Merge transformation
  - Sorter transformation
  - Stored Procedure transformation with exception handling
  - Transaction Control transformation
  - Web Services consumer transformation
  - XML Generator transformation with flat file or database sources
  - XML Parser transformation with flat file or database sources
- If you use a mapplet in a synchronization task, the PowerCenter mapplet cannot contain multiple Input transformations.
- If you use a mapplet in a mapping task, the PowerCenter mapplet can contain multiple Input transformations.
- Data Integration flattens PowerCenter mapplets with multiple input groups into mapplets with one input group. Therefore, the ports in each input group in the PowerCenter mapplet must have unique names. If the names are not unique, rename the input ports in PowerCenter before you export the PowerCenter XML file that contains the mapplet.
- The PowerCenter mapplet cannot contain reusable objects such as shortcuts because Data Integration does not use a repository to store reusable objects. Export the mapplet without reusable objects.

PowerCenter mapplets in Data Integration tasks

Use the following rules and guidelines for using PowerCenter mapplets in Data Integration tasks:

- You can add Lookup transformations between the source and a mapplet.
- You can add Expression transformations between the source and a mapplet, and between a mapplet and the target.
- When you add a mapplet to a synchronization task, the synchronization task wizard removes existing field mappings. The synchronization task wizard doesn't remove existing field mappings if you add a passive mapplet between the source and target.
- When you use an active mapplet in a synchronization task that includes a saved query, the synchronization task ignores the configured target option for the task and inserts data to the target.
- Data Integration retains PowerCenter session-level overrides to the mapping.

Configuring a PowerCenter mapplet

To create a mapplet in Data Integration, you upload a PowerCenter XML file that contains a PowerCenter mapplet.

When you upload the PowerCenter XML file, Data Integration creates a mapplet based on the mapplet definition in the XML file.

1. To create a mapplet, click **New** > **Mapplets** > **Mapplet - PC Import** and then click **Create**.

   To edit a mapplet, on the **Explore** page, navigate to the mapplet. In the row that contains the mapplet, click **Actions** and select **Edit**.
2. Configure the following details:

Detail	Description
Mapplet Name	Name of the mapplet. Mapplet names can contain alphanumeric characters, spaces, and the following special characters: _ . + - Maximum length is 100 characters. Mapplet names are not case sensitive.
Location	Project folder in which the mapplet is to reside.
Description	Description of the mapplet. Maximum length is 255 characters.
Mapplet Type	Whether the mapplet is active or passive. The mapplet type depends on the type of PowerCenter transformations in the mapplet: - Active. The mapplet includes one or more active PowerCenter transformations. - Passive. The mapplet includes only passive PowerCenter transformations.

3. To upload the PowerCenter XML file, click **Upload**.
4. In the **Upload Mapplet XML File** dialog box, click **Choose File**.
5. Browse to the appropriate location and select the PowerCenter XML file.
6. Click **OK**.

   The Mapplet XML File Details area shows the connections, input fields, and output fields.
7. Click **Save**.

CHAPTER 12

Saved queries

A saved query is a component that you create to run SQL statements against a database. You can use a saved query as the source object in a synchronization task or as the query in an SQL transformation.

You can use a saved query in the following places:

As the source in a synchronization task

Create a saved query when you want to use a database source that you cannot configure using the single- or multiple-object source options. For example, you might create a saved query to include source filters, or to perform a complicated join of multiple tables. The query must be a SELECT statement.

To use a saved query in a synchronization task, first create the saved query component. Then when you configure the synchronization task, select the saved query as the source object. You can add one saved query to a synchronization task.

As the query in an SQL transformation in a mapping

You can create a saved query to use as the query in an SQL transformation. The query can contain one or more SQL statements. You can use one saved query in an SQL transformation.

To use a saved query in an SQL transformation, first create the saved query component. Then when you configure the SQL transformation in a mapping, you select the saved query to use.

To use saved queries, your organization must have the appropriate license. For more information, contact Informatica Global Customer Support.

Saved query syntax

When you create a saved query, enter an SQL statement that is valid for the database that you want to run the query against.

You can use different SQL statements based on where you use the saved query:

Synchronization task

When you create a saved query to use as the source in a synchronization task, the SQL statement must be a SELECT statement. Data Integration uses the SQL statement to retrieve source column information. You can edit the data type, precision, or scale of each column before you save the saved query.

For example, you might create a saved query based on a TRANSACTIONS table that includes transactions from 2016 with the following SQL statement:

```
SELECT TRANSACTION_ID, TRANSACTION_TOTAL, TRANSACTION_TIMESTAMP from
dbo.TRANSACTIONS WHERE TRANSACTION_TIMESTAMP>'0:0:0:0 01/01/2016'
```

Data Integration ensures that saved query column names are unique. If an SQL statement returns a duplicate column name, Data Integration adds a number to the duplicate column name as follows:

<column_name><number>

SQL transformation

When you create a saved query to use in an SQL transformation, you can use one or more of the following SQL statements in the query:

ALTER

CALL

COMMENT

COMMIT

CREATE

DELETE

DROP

EXPLAIN PLAN

GRANT

INSERT

LOCK TABLE

MERGE

RENAME

REVOKE

ROLLBACK

SELECT

TRUNCATE

UPDATE

Use the following guidelines when you create the query:

- You can use aggregate functions such as COUNT with Salesforce connections only.
- Do not use conversion functions, such as TO_CHAR or TO_DATE.
- Do not use an asterisk (*) to select all columns of a table. List the columns that you want to select.
- Do not use SQL transformation parameter binding or string substitution notation such as ?input_field? or ~input_field~ in the query. Saved queries have no information about SQL transformation input fields.

Tip: Test the SQL statement you want to use on the source database before you create a saved query. Data Integration does not display specific error messages for invalid SQL statements.

Using saved queries in synchronization tasks

Use the following rules and guidelines to use a saved query in a synchronization task:

- You can add one saved query to each synchronization task.
- You cannot delete a saved query that is used in a synchronization task.

- If you edit column information for a saved query, the synchronization task does not write error rows to the error rows file at runtime.
- When you use an active mapplet with a synchronization task that includes a saved query, the synchronization task inserts data to the target even if you select a different target option.

Using saved queries in SQL transformations

Configure an SQL transformation to use a saved query on the **SQL** tab of the **Properties** panel. You can use one saved query in an SQL transformation.

1. On the **SQL** tab of the **Properties** panel, select the connection that contains the tables that you want to run the query against.
2. Set the SQL type to **SQL Query**.
3. Set the query type to **Saved Query**.
4. Select the saved query that you want the SQL transformation to process.

For more information about SQL transformations, see *Transformations*.

Configuring a saved query

Create a saved query that you can use as a source object in synchronization tasks or as a query in SQL transformations.

1. Click **New** > **Components** > **Saved Query** and then click **Create**.

   To edit a saved query, on the **Explore** page, navigate to the saved query. In the row that contains the saved query, click **Actions** and select **Edit**.
2. Enter the following details:

Detail	Description
Saved Query Name	Name of the saved query. Query names can contain alphanumeric characters, spaces, and the following special characters: _ . + - Maximum length is 100 characters. Query names are not case sensitive.
Location	Location of the saved query. Browse to the folder where you want to store the saved query or use the default location. If the **Explore** page is currently active and a project or folder is selected, the default location for the asset is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.
Description	Description of the saved query. Maximum length is 255 characters.

Detail	Description
Database Type	Source database type. Select one of the following database types: - Salesforce - Oracle - SQL Server - MySQL - ODBC - MS Access To use the query in an SQL transformation, the database type must be Oracle or SQL Server.
SQL Query	SQL statements that make up the SQL query. To use the query as the source in a synchronization task, enter a SELECT statement. Data Integration uses the SELECT statement to retrieve the source column information. To use the query in an SQL transformation, enter one or more valid SQL statements. Do not use SQL transformation parameter binding or string substitution notation in the query because saved queries have no information about SQL transformation input fields.

3. If you enter a SELECT statement, click **Get Columns** and select a connection.

   The Saved Query Column Details table displays the columns selected in the SQL statement.
4. Optionally, in the Saved Query Column Details table, edit the data type, precision, or scale.

   If you edit these values, Data Integration does not write error rows into the error rows file.
5. Click **Save**.

CHAPTER 13

Shared sequences

Shared sequences are reusable sequences that you can use in multiple Sequence transformations. When you use a shared sequence, the Sequence transformation uses the properties of the shared sequence to generate values. Use a shared sequence when you want to assign numeric values within the same sequence to data in multiple mapping tasks.

Multiple mappings and mapplets can use the same shared sequence. When you run the mapping task, Data Integration reserves a set of values in the sequence so that each mapping task generates unique values.

For example, you want to assign a unique ID to entries in your customer tables. You have two mappings that load customer data to the same target table, and the mappings are scheduled to run at the same time. Use a shared sequence to ensure that you do not get duplicate IDs in the target.

You create shared sequences on the **New Assets** page. You can create non-shared sequences in a Sequence transformation. For more information about creating non-shared sequences, see *Transformations*.

You can't use shared sequences in mappings in SQL ELT mode.

Shared sequence properties

When you create a shared sequence, you define general properties such as the sequence name, and properties that define the sequence such as initial value, increment value, and end value.

The following table lists the properties you define for a shared sequence:

Property	Description
Name	Required. Name of the shared sequence. Can contain alphanumeric characters and underscores (_). Maximum length is 200 characters.
Location	Project or folder in which the sequence resides. If the **Explore** page is currently active and a project or folder is selected, the default location for the asset is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.
Description	Optional. Description for the shared sequence.
Increment By	The difference between two consecutive values in a generated sequence. For example, if Increment By is 2 and the existing value is 4, then the next value generated in the sequence will be 6. Default is 1. Maximum is 2,147,483,647.

Property	Description
End Value	Maximum value that the mapping task generates. If the sequence reaches this value during the task run with rows left to process and the sequence is not configured to cycle, the run fails. If the sequence is configured to cycle, the sequence uses this value and then starts over at the Cycle Start Value. Maximum is 9,223,372,036,854,775,807.
Initial Value	The value you want the mapping task to use as the initial value in the sequence the first time that the sequence is used. If you want to cycle through a series of values, this value must be greater than or equal to the Cycle Start Value and less than the End Value. If you reset the shared sequence, the sequence is reset to this value. Default is 0.
Cycle	If enabled, the mapping task cycles through the sequence range. If disabled, the task stops the sequence at the configured End Value. The session fails if the task reaches the End Value and still has rows to process. Default is disabled.
Cycle Start Value	Start value of the generated sequence that you want the mapping task to use when the sequence is configured to cycle. When the sequence reaches the end value, it cycles back to this value. Default is 0.
Number of Reserved Values	The number of sequence values the mapping task caches at one time. Must be greater than or equal to 1000.
Current Value	Displays the current start value in the sequence.

Number of reserved values

The number of reserved values determines the number of values in the sequence that the mapping task caches at one time during each task run.

When multiple Sequence transformations use the same shared sequence, multiple instances of the shared sequence can exist at the same time. To avoid generating duplicate values, reserve a range of sequence values for each mapping by configuring the number of reserved values.

For example, you configure a shared sequence as follows: Number of Reserved Values = 1000, Initial Value = 1, Increment By = 1. Two mappings use the shared sequence. When the first mapping task runs, Data Integration reserves 1000 values (1-1000) for the mapping task and updates the current value in the repository to 1001. When the second mapping task runs, Data Integration reserves the next 1000 values (1001-2000) and updates the current value to 2001. When either mapping task uses all of its reserved values, Data Integration reserves a new set of values and updates the current value.

The number of values that Data Integration reserves in a sequence is determined by multiplying the Number of Reserved Values by the Increment By value.

For example, if you have a shared sequence that begins at 2, increments by 2 and the number of reserved values is 1000, Data Integration reserves 1000 values in the sequence, or numerical values 2-2002, and updates the current value to 2003.

By default, the number of reserved values is 1000. To increase performance, you can increase the number of reserved values. This reduces the number of calls that Data Integration must make to the repository. The number of reserved values cannot be less than 1000.

Values that are reserved for a task run but not used in the task are lost when the task completes. For example, you generate IDs in a table with 1250 rows. You configure the shared sequence to generate values in increments of 1 and set the number of reserved values to 2000. When the task runs, Data Integration reserves 2000 values for the task and updates the current value to 2001. When the next task runs, the sequence begins at 2001, and values 1251-2000 are lost.

Note: If you use partitions, Data Integration reserves a set of values for each partition.

Creating a shared sequence

Create a shared sequence on the **New Asset** page.

1. Click **New** > **Components** > **Shared Sequence** and then click **Create**.
2. Define the sequence properties.
3. Save the sequence.

Using a shared sequence

You can use a shared sequence in a Sequence transformation in a mapping or mapplet.

1. Open the mapping or mapplet.
2. Add a Sequence transformation to the canvas and connect it to a downstream transformation.
3. On the **Sequence** tab, select **Use Shared Sequence**.
4. Select the sequence to use.
5. Configure the Sequence advanced properties and output fields.
   For more information about configuring a Sequence transformation, see *Transformations*.

Note: If you don't select **Use Shared Sequence**, the Sequence transformation generates a non-shared sequence.

Resetting a shared sequence

Reset a shared sequence to update the current value of the sequence to the defined initial value.

1. Open the shared sequence.
2. Click **Edit**.
3. Click **Reset Current Value** in the **Sequence Properties** area.
4. Save the shared sequence.

CHAPTER 14

User-defined functions

User-defined functions are reusable functions that you can use in expressions. Create user-defined functions to build complex expressions using the Informatica Intelligent Cloud Services transformation language. User-defined functions use the same syntax and can use the same transformation language components as transformation and field expressions.

You can include a user-defined function in a transformation expression in a mapping or mapplet, in a field expression in a mapping task, or in another user-defined function. To use a user-defined function in an expression, you select the user-defined function in the expression editor and enter the required arguments.

You can't use a user-defined function in an expression in a synchronization task.

You can include user-defined functions in the following transformations in advanced mode:

- Aggregator
- Expression

You can't use user-defined functions in expressions in mappings in SQL ELT mode.

Example

You want to remove leading and trailing spaces from a text string such as a name or address. You create a user-defined function called RemoveSpaces that takes a text string as an argument and performs the LTRIM and RTRIM functions. When you configure the user-defined function, you enter the following expression:

```
LTrim( RTrim(TextString) )
```

After you create the function, you use it in an Expression transformation to remove leading and trailing spaces from the incoming field, LAST_NAME. The expression field contains the following expression:

```
:UDF.RemoveSpaces(LAST_NAME)
```

In the expression, the user-defined function name is prefaced with :UDF. The incoming field LAST_NAME is passed to the function as the argument.

Creating user-defined functions

Create a user-defined function on the **New Asset** page. The user-defined functions that you create are available to all users in your organization based on their privileges.

1. Click **New** > **Components** > **User-Defined Function** and then click **Create**.
2. Configure general properties such as the function name, location, and return type.
3. Optionally, create arguments for the function.

When you create arguments, configure the name, data type, and description for each argument.

4. Create the function expression.
5. Validate and save the function.

User-defined function general properties

When you create a user-defined function, you must define general properties such as the function name and return type. Define general properties on the **General** tab.

The following table describes the properties:

Property	Description
Name	Name of the function. Must be unique within an organization. The name must begin with a letter and can contain letters, numbers, and the following special characters: _ @ $ # The name cannot exceed 100 characters and cannot contain spaces.
Location	Location of the user-defined function. Browse to the folder where you want to store the user-defined function or use the default location. The default location is the location of the most recently saved asset.
Description	Description of the user-defined function. The description appears in the expression editor when you select the function in a transformation expression, a field expression, or another user-defined function.
Return Type	Data type of the values that the function returns. Can be binary, date, numeric, or string.

User-defined function arguments

When you create a user-defined function, you can include arguments to define the values that you pass to the function. Create arguments on the **Arguments** tab. You can create up to 10 arguments.

To create an argument, click **Add**, and enter the following information:

Property	Description
Name	Name of the argument. Must be unique within the function. The name must begin with a letter and can contain letters, numbers, and underscore characters (_). It cannot exceed 100 characters and cannot contain spaces.
Type	Data type of the argument. Can be binary, date, numeric, or string.
Description	Description of the argument. The description appears in the Arguments and Functions area on the **Expression** tab when you select the argument. It also appears in the expression editor when you use the function in a transformation or field expression.

Arguments are passed to the function in the order in which they appear on the **Arguments** tab. To rearrange the order, select an argument and click **Move up** or **Move down**.

To delete an argument, click **Delete** in the row that contains the argument.

User-defined function expression

Create the expression that defines the function on the **Expression** tab. The function expression can contain constants, operators, built-in functions, and other user-defined functions. You can create a complex expression by nesting functions within functions.

To create an expression, enter the expression in the expression editor. The expression that you create must follow the same guidelines and use the same syntax as any transformation or field expression. For more information about creating expressions, see *Tasks*.

You can add arguments, built-in functions, and other user-defined functions to the expression by selecting them in the Arguments and Functions area of the expression editor and clicking **Add**. Alternatively, press the **Ctrl** + **Space** keys to see a list of recommended arguments and functions in-line. You can also type in the expression manually.

An expression cannot contain cyclic references. For example, the expression for user-defined function Function_A cannot include Function_A. Also, if user-defined function Function_A calls user-defined function Function_B, then the expression for Function_B cannot call Function_A.

To validate the expression, click **Validate**. Data Integration validates the expression. It does not validate the user-defined functions and expressions that use the function.

Editing and deleting user-defined functions

Certain restrictions apply when you edit or delete a user-defined function.

Editing a user-defined function can affect the expressions and functions that use it. Consider the following guidelines when you edit a user-defined function:

- If you edit a user-defined function and the updated function is valid, Data Integration propagates the changes to all expressions and user-defined functions that use the function.

  Sometimes this can make the expressions and functions that use the user-defined function invalid. For example, you edit a function, change the number of arguments from one to two, and validate the function. Even though the updated function is valid, expressions and functions that use the user-defined function become invalid if they pass one argument to the function.
- If you edit and save a user-defined function but the updated function is not valid, Data Integration does not propagate the changes to the expressions and user-defined functions that use it.
- You cannot rename a user-defined function after you save it.

To delete a user-defined function, you must first remove it from all expressions and functions that use it. You cannot delete a user-defined function that is used in an expression or in another user-defined function.

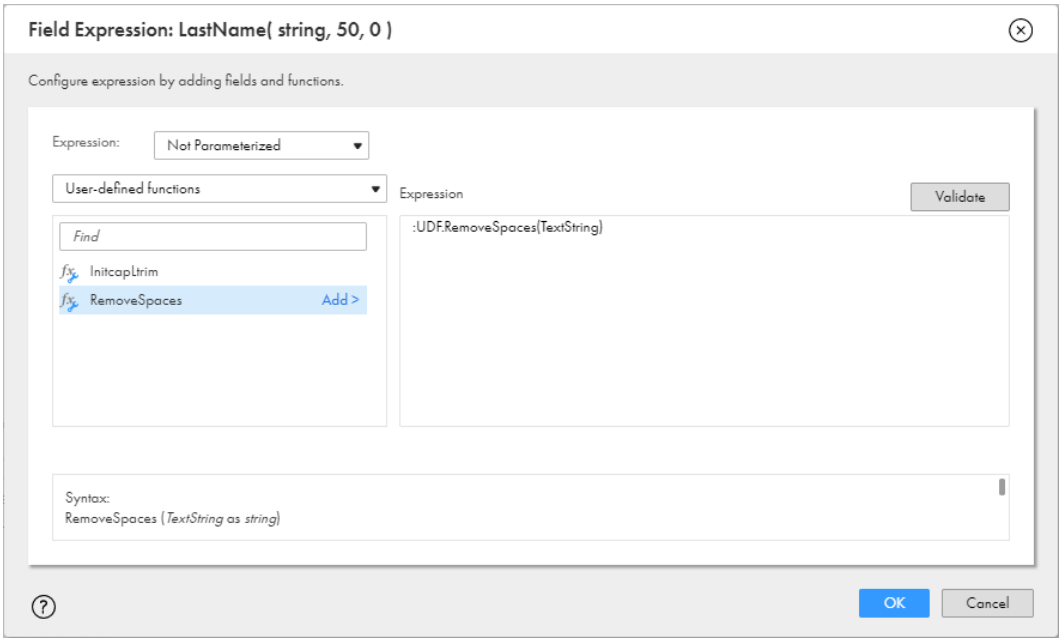
Tip: View the dependencies for the user-defined function to find the assets that use the function.

Creating expressions with user-defined functions

You can add a user-defined function to a transformation or field expression.

When you create an expression, valid user-defined functions appear in the expression editor. If you type in an expression manually, precede each user-defined function with :UDF.

The following image shows a user-defined function selected in the expression editor of an Expression transformation:



When you select a user-defined function, the expression editor shows the function syntax in the following format:

```
<function name> (<argument 1> as <data type>, <argument N> as <data type>)
```

For example:

```
RemoveSpaces(TextString as string)
```

When you add the function to the expression, the function includes the prefix :UDF, as shown in the following example:

```
:UDF.RemoveSpaces(TextString)
```

After you add the function to the expression, replace the arguments with field names or in-out parameters. Don't include the table name in the argument. Use only the field name as shown in the following example:

```
:UDF.RemoveSpaces(NAME)
```

For more information about creating expressions, see *Tasks*.

When you validate the expression, Data Integration doesn't validate the user-defined function. It only validates the expression.

Parameterizing user-defined functions

You can use an in-out parameter that is configured as an expression variable to fully or partially parameterize a user-defined function in an expression.

To parameterize a user-defined function, create a string in-out parameter and enable **Is expression variable**. To fully parameterize the user-defined function, use the parameter in place of the entire user-defined function. To partially parameterize the user-defined function, use the in-out parameter in place of the user-defined function arguments. At run time, resolve the parameter in the mapping task or parameter file.

For more information about in-out parameters, see *Mappings*.

Full parameterization

Use full parameterization when you want to parameterize the user-defined function in an expression. When you use an in-out parameter in place of a user-defined function, parameter names are case sensitive. When you resolve the parameter, include the :UDF prefix.

For example, you have the in-out parameter $$UDFparameter and you use it to fully parameterize a user-defined function. In the parameter file, you resolve the parameter with the following function:

```
$$UDFparameter=:UDF.RemoveSpaces(names)
```

Partial parameterization

Use partial parameterization when you want to parameterize the field name in the argument, or when you want to parameterize a nested user-defined function. When you parameterize a nested user-defined function, the parameter in the function argument must be in capital letters. You can nest one user-defined function.

For example, you create the in-out parameter $$UDF and configure it as an expression variable. In the expression, you configure the following function:

```
:UDF.RemoveSpaces($$UDF)
```

In the parameter file, you resolve the parameter with the following function:

```
$$UDF=:UDF.Replace_Chars_special(Name)
```

Full and partial parameterization

You can use both full and partial parameterization within the same function. Use both full and partial parameterization when you want to parameterize the user-defined function and the function arguments.

For example, you have the in-out parameter $$UDF and use it in an expression to fully parameterize a user-defined function. In the parameter file, you resolve the parameter with the following values:

```
$$UDF=:UDF.Replace_Chars_special($$Field)
$$Field=Name
```

When you want to pass either the CurrentTaskName, CurrentRunId, or SESSSTARTTIME system variable as an argument in a fully parameterized user-defined function, you must pass the system variable through another in-out parameter.

For example, you want use CurrentTaskName in a parameterized user-defined function. You use the in-out parameter $$UDF to fully parameterize the function. In the parameter file, you resolve the parameter with the following values:

```
$$UDF=:UDF.udf_InitcapLtrim($$tasknameparam)
$$tasknameparam='$CurrentTaskName'
```

Data classifications

If your organization uses Metadata Command Center, you can link user-defined functions to data classifications that are associated with objects of a catalog source in Metadata Command Center. When you use the source object in a mapping, CLAIRE recommends user-defined functions based on the data classifications associated with the object columns.

Before you can link data classifications to user-defined functions, in Metadata Command Center, create the data classifications, associate them with a catalog source, and execute a data classification scan job . In Data Integration, associate the data classifications with user-defined functions so that CLAIRE recommends the functions when you use source objects that are also associated with the data classification.

Adding data classifications

Add data classifications to a user-defined function so that CLAIRE recommends the function in mappings when you add a source object with a matching data classification.

1. On the **Explore** page, navigate to the user-defined function.
2. In the row that contains the function, click **Actions** > **Edit Data Classifications**.
3. In the **Data Classifications** window, drag and drop the data classifications that you want to associate with the user-defined function.
4. Click **Save**.

INDEX

A

active mapplets
  description 81, 88
advanced mode
  intelligent structure model 52
asset components 9

B

B2B Gateway
  intelligent structure model 53
business services
  defining 13

C

Cloud Application Integration community
  URL 7
Cloud Developer community
  URL 7
components
  business services 13
  file listeners 15, 18
  file listeners in taskflows 17
  fixed-width file formats 29
  hierarchical schemas 41
  industry data service customizer 43
  intelligent structure models 51
  mapplets 81
  saved queries 91
  shared sequence 95

D

data engineering
  intelligent structure model 53
Data Integration community
  URL 7

E

exception handling
  in mapplet stored procedures 88

F

file listeners
  configuring 19
  invoking taskflows 17
  orchestrating taskflow execution 17
file listeners in partner flows 18
file listeners in taskflows 17
fixed-width file formats
  configuring 29
flat files
  fixed-width file formats 29

H

hierarchical mappers 33
hierarchical schemas
  creating 42
hierarchical to hierarchical 33
hierarchy builder transformation
  intelligent structure model 52

I

industry data service customizer
  adding an element 45
  customizing a message 44
  deleting an element 46
  editing an element 45
  global settings 44
  HIPAA 46
  HL7 48
  message definition 43
  message properties 46
  message structure 43
  positional settings 44
Informatica Global Customer Support
  contact information 8
Informatica Intelligent Cloud Services
  web site 7
intelligent structure
    data drift
      intelligent structure 60
  document identifiers 75
  editing 71, 72, 75, 77
  example 63
    model input
      intelligent structure 54
  refining 67
  repeating groups 57, 75
  viewing 77
intelligent structure model
  advanced mode 52
  B2B Gateway 53
  creating 61
  data engineering 53
  development 53
  discovery process 53
  editing 76
  exporting 63

intelligent structure model *(continued)*
hierarchy builder transformation 52
prefix 76
scenario 64
structure parser transformation 52
suffix 76
intelligent structure models 51
intelligent structure nodes
editing 72
intelligent structures
actions 76
foreign keys
description 59
nodes 76
primary and foreign keys 59

M

maintenance outages 8
mapplet
output 81
Mapplet
input 81
mapplet input 82
mapplet output 82
mapplet parameters 83
mapplets
active and passive 81, 88
configuring 90
overview 81
rules for PowerCenter XML files 88
rules for using in tasks 89
Mapplets
PowerCenter 87

N

number of reserved values 96

O

output groups
editing 71

P

parameters
in mapplets 83
passive mapplets
description 81, 88
PowerCenter mapplets 87
PowerCenter XML files
rules for mapplets 88
primary keys
description 59
properties
shared sequence 95

R

relational output
working with 71
repeating groups
description 57
working with 75

S

saved queries
configuring 93
creating 91
overview 91
using in synchronization tasks 92
shared sequence
creating a shared sequence 97
number of reserved values 96
resetting a shared sequence 97
sources
custom saved queries 91
SQL transformations
saved queries 91
status
Informatica Intelligent Cloud Services 8
stored procedures
exception handling in mapplets 88
structure parser transformation
intelligent structure model 52
system status 8

T

trust site
description 8

U

upgrade notifications 8
user-defined functions
arguments 99
creating 98
deleting 100
editing 100
expression 100
general properties 99
overview 98
return type 99

W

web site 7