ptc

thingworx® analytics™

ThingWorx Analytics Server Transition Guide to 8.1

8.1

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Document Revision History

Revision Date	Description of Change
September 29, 2017	Initial document version for Release 8.1.

Note

Between software releases, check the Reference Document page of the PTC Support Portal for updated versions of this document.

Contents

1

Overview of Changes

This guide provides information for users of previous ThingWorx Analytics versions who want to upgrade to the new microservice architecture of the 8.1 release. The new architecture provides near functional parity with the old architecture but will require changes in the way you interact with your ThingWorx Analytics Server.

The guide provides some high-level introductory information about the new architecture, some instructions for preparing your data and creating a dataset, and some reference mapping sections to help you navigate between the legacy REST endpoints and the new microservices.

The following subsections describe some of the changes to ThingWorx Analytics Server.

The Move to Microservices

In this release, the ThingWorx Analytics monolithic server has been divided into multiple microservices. A microservice is an independent process that can focus on a specific area of functionality and run independently of other microservice processes. This new architecture provides a lightweight application that is more robust when deployed and simpler to upgrade. The transition to the microservices architecture is part of long-term strategy to increase scalability and support high availability installations.

Integration with ThingWorx Core

In this release, ThingWorx Analytics functionality has become a native part of the ThingWorx Core application. A number of system-level Thing Templates and Data Shapes have been added to ThingWorx Core in support of ThingWorx Analytics. These are available out-of-the-box when ThingWorx Core is installed, but are only useful when a ThingWorx Analytics server is connected.

Thing Templates

For each ThingWorx Analytics Server deployment, the Thing Templates are used to automatically instantiate a set of Analytics Things, one for each microservice. The Analytics Things provide functionality via services that are available on each Analytics Thing. These services include submitting jobs, querying (jobs, statuses, results), retrieving results, and managing tags on results and statuses. As a result, Analytics Server APIs are now accessible in the same way that any ThingWorx service can be accessed (for example in a ThingWorx Composer mashup or via a ThingWorx REST API).

Note

There is no longer a direct ThingWorx Analytics Server REST API. Support for accessing the services via REST calls is now provided through the ThingWorx Core REST API layer. A new URI pattern is required to do so.

Data Shapes

In ThingWorx Core, a Data Shape is a way of defining a set of data so that it can be more easily consumed. ThingWorx Core supports a number of Data Shape base types, but the infotable is the type ThingWorx Analytics Server is now using to handle both data inputs and outputs. Instead of referencing a dataset in the URL of an API call, the dataset information is now provided as an input parameter in the request body. Two key Data Shapes for referencing datasets are the following:

- AnalyticsDatasetRef – An infotable that references a specific dataset. It includes a URI and the data format (both will vary depending on whether the data is a stored dataset or in-place data provided with a request).
- AnalyticsDatasetMetadata – An infotable that contains the machine learning characteristics that describe a dataset. It includes fieldName, dataType, opType, and more.

For more detailed information about each of these infotables, see Key Analytics Infotables.

ThingWorx Core Integration and Multi-tenancy

Another result of the integration with ThingWorx Core, is that the concept of multi-tenancy has changed. All data, services, and results are globally available within a specific Analytics Server deployment. All users, with access to the

Things that represent the server deployment, can access any dataset, service, job, or result. To restrict access to these objects, multiple Analytics Server deployments are necessary. There are two ways to set up multiple deployments:

- Multiple deployments that connect to a single ThingWorx Core server – In this scenario, use the ThingWorx Core permissions and visibility functionality to restrict access, to the Things associated with each deployment, to specific users, groups, and organizations.
- Multiple deployments that each connect to a different ThingWorx Core server – In this scenario, there should be a one-to-one correspondence between the Analytics server and the ThingWorx Core server. Users would only be able to access the Things associated with the deployment they are authenticated through.

Other Aspects of the ThingWorx Core Integration

ThingWorx Analytics Server can leverage several useful aspects of ThingWorx Core functionality, such as:

- Application keys – No longer need the ThingWorx Analytics Server App ID/ Key combination for access.
- Access and permissions – Can be managed through the ThingWorx organizations, user groups, and users.
- Tomcat – Runs on the ThingWorx Core Tomcat server, so no Analytics-specific installation of Tomcat is necessary.
- ThingWorx Composer – Can be used as a front end to interact with microservice things, services, and file repositories.

> **Note**
>
> The ThingWorx Composer comes in two flavors, classic and New Composer. This guide assumes the New Composer interface is in use. For information about how to switch to this view, see New Composer in the ThingWorx Core Help Center

Changes to Data Loading and Storage

Data storage will no longer require installation of a PostgreSQL database. Instead, uploaded CSV data is converted to an optimized Parquet format and stored directly in the file system. This change removes the limitations on the number of data columns the system can handle. It also allows for streamlining the dataset creation process. Since the data is converted to a Parquet format, there is no need to separately optimize the dataset. Even when new data is appended to an existing dataset, a new partition is added and reoptimization is not required.

In addition, data access is now URI-driven and can be accessed from a variety of locations:

- AnalyticsUploadStorage – a ThingWorx file repository where data can be uploaded and stored (`thingworx://AnalyticsUploadStorage/`)
- ThingWorx Analytics Server file system – data can be loaded directly from a location that is readable by the Analytics Server and accessed in place, useful for small datasets with rapidly changing data (`file:/`)
- API request body – data can be supplied as part of an API call, useful for scoring or model evaluation jobs or for appending small amounts of data to an existing dataset, this data is not stored (`body:/`)
- ThingWorx Analytics Server dataset – a dataset created in ThingWorx Analytics (`dataset:/<jobID>`)

Less Dataset-centric

The dependence on a specific dataset has been removed for much of ThingWorx Analytics Server functionality. Models can be trained and scored based on any dataset that is structured appropriately. Signals, profiles, and clusters can also be generated without reliance on a specific dataset. In Analytics Builder, this change also means that generating signals and profiles is not dependant on first building a predictive model.

This change allows datasets and models to be managed independently of each other. It provides more flexibility in handling data and querying the dataset.

Functionality Not Included

In the 8.1 release of ThingWorx Analytics Server, the following functionality is not included:

- Prescriptive batch scoring (real-time scoring has been introduced)
- Distributed installation
- Data Connect

In each case, the excluded functionality will be re-introduced in a subsequent release. In the meantime, users of these items of functionality can continue to use the 8.0 release.

Note

The 8.0 release cannot access datasets in the 8.1 system.

2

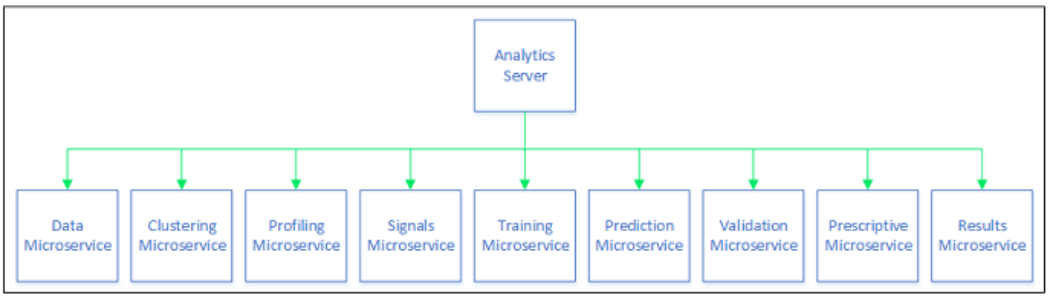
Introduction to the Microservice Architecture

In the 8.1 release of the ThingWorx Analytics Server, the classic server monolith has been replaced by a series of independent microservices. This new structure groups services around specific elements of functionality (data, training, results). The new architecture provides a more robust and stable deployment. Services are executed by job type so that multiple microservices can run in parallel but problems in one microservice do not interrupt performance in others. The microservice architecture also provides the opportunity for ThingWorx Analytics developers to isolate areas of functionality for research and development.

In addition to restructuring the Analytics Server, its capabilities are being made natively available in the ThingWorx Core software. Instead of interacting directly with the Analytics Server, the APIs are now available as services on a series of ThingWorx Core Things. Each Thing represents a connected microservice.

To use the Analytics functionality, both ThingWorx Core and ThingWorx Analytics Server must be installed, preferably on separate servers. When both servers are running, an edge agent acts as an integration point between the two servers. It automatically registers with the ThingWorx server and instantiates the Things that represent the connected microservices.

The microservices are organized in a functional hierarchy that informs which specific services are available to each microservice. The high-level diagram below shows the available microservices. For more information about where specific services are available, see the mapping section: Mapping Services to REST APIs.


Analytics Server
Data Microservice
Clustering Microservice
Profiling Microservice
Signals Microservice
Training Microservice
Prediction Microservice
Validation Microservice
Prescriptive Microservice
Results Microservice

3

Prerequisites

Before you can use ThingWorx Analytics, the following necessary prerequisites must be met:

- ThingWorx Core 8.1 – Unlike past ThingWorx Analytics releases, ThingWorx Core must be installed and running first, preferably on a separate server. For specific prerequisites and installation information, see *Installing ThingWorx Core* available in the Reference Documents of the PTC eSupport portal.
  - An application key must be created in ThingWorx and will be required during installation of ThingWorx Analytics Server (For instructions, see Application Keys in the ThingWorx Core Help Center.)
  - Network connectivity must be available between ThingWorx and the ThingWorx Analytics Server.
- ThingWorx Analytics Server 8.1 – In this release, installation is available via either Docker (for Windows or Linux) or native Linux services. For specific prerequisites and installation information, see the Docker and native Linux installation guides available in the Reference Documents of the PTC eSupport portal.
  - As a native part of ThingWorx Core, Analytics-specific installation of Tomcat is no longer necessary.
  - Also as part of ThingWorx Core, data storage is no longer handled in a database. Loaded data is converted to a Parquet format and stored in the file system. Therefore, PostgreSQL installation is no longer required.

4

Security Considerations

When deploying a ThingWorx Analytics Server, especially in a production environment, be sure to observe all best practice security measures. Before making the server or its components accessible to other users, consider the following security measures:

- Change default passwords
- Disable root login for SSH access
- Install an IP address-filtering firewall

All datasets, services, and results are available to all users with access to a specific ThingWorx Analytics Server deployment. Any user, with access to the Things that represent the server deployment, can access any dataset, service, job, or result. To restrict access to these objects, multiple Analytics Server deployments are necessary. There are two ways to set up multiple deployments:

- Multiple deployments that connect to a single ThingWorx Core server – In this scenario, use the ThingWorx Core permissions and visibility functionality to restrict access, to the Things associated with each deployment, to specific users, groups, and organizations.
- Multiple deployments that each connect to a different ThingWorx Core server – In this scenario, there should be a one-to-one correspondence between the Analytics server and the ThingWorx Core server. Users would only be able to access the Things associated with the deployment they are authenticated through.

5

Prepare Data and Metadata

As in previous releases, preparing data for ThingWorx Analytics includes both a CSV file containing the raw data and a JSON file defining the metadata structure of the data. Keep in mind the following when preparing these files for upload:

- In the CSV file, make sure there are no spaces before or after the column header names.
- The JSON format for the metadata file has changed. Some parameters have been removed from the metadata (such as whether a feature can be designated as a goal or lever variable). Other parameters have been added (such as opType and time series fields). The following chart outlines the new metadata format. A metadata sample follows the chart.

Note

Optional parameters can be set to null or omitted (both will have the same effect).

Parameter	Description	Required/ Optional
fieldName	The exact name of the field as it appears in the dataset.	Required
values	A list of the acceptable values for the field. **Note** For Ordinal opTypes, the values must be presented in the correct order.	Required if the opType is Ordinal Optional for Categorical opType

Parameter	Description	Required/ Optional
		Do not use for Boolean and Continuous
range	For a Continuous field, defines the minimum and maximum values the field can accept. For informational purposes only.	Optional
dataType	Describes what type of data the field contains. Options include: Long, Integer, Short, Byte, Double, Boolean, String, Other. **Note** Select the most accurate dataType. Selecting the String dataType for numeric data can lead to undesirable results.	Required
opType	Describes how the data in the field can be used. Options include: Categorical, Boolean, Ordinal, Continuous, Informational, Temporal, Entity_ID	Required
timeSamplin-gInterval	An integer representing the time between observations in a temporal field.	Required if the opType is Temporal Do not use for other opTypes
isStatic	A flag indicating whether or not the value in a temporal field can change over time. Marking a field as static reduces training time by removing redundant data points for fields that do not change.	Optional

Metadata Sample:

```
[
  {
    "fieldName": "Age Group",
    "values": [
      "<=5",
      "6-11",
```

```
        "12-18",
        "19-25",
        "26-40",
        "41-54",
        "55-64",
        "65-74",
        "75-84",
        "85+"
      ],
      "range": null,
      "dataType": "STRING",
      "opType": "ORDINAL",
      "timeSamplingInterval": null,
      "isStatic": false
    },
    {
      "fieldName": "Annual Well Visit Claims",
      "values": [
        "Not Completed",
        "Completed"
      ],
      "range": null,
      "dataType": "STRING",
      "opType": "ORDINAL",
      "timeSamplingInterval": null,
      "isStatic": false
    },
  {
      "fieldName": "AsthmaFlag",
      "values": null,
      "range": null,
      "dataType": "STRING",
      "opType": "BOOLEAN",
      "timeSamplingInterval": null,
      "isStatic": false
    }
   ]
```

6

Upload the Data

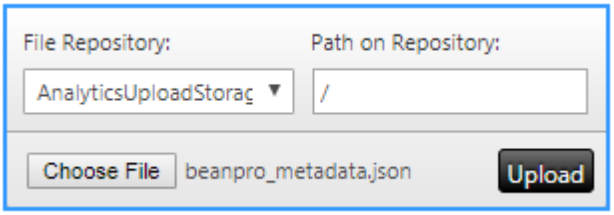
Now that ThingWorx Analytics functionality is accessible from ThingWorx Core, the CSV data and JSON metadata files both need to be uploaded to an **AnalyticsUploadStorage** Thing in the ThingWorx Core. There are multiple procedures available for uploading these files.

- Upload via the **File Upload** widget in a mashup
- Upload via a REST client
- Upload via Analytics Builder
- Upload Large Volume Data Directly

Upload via the File Upload Widget

Upload data and metadata files via the **File Upload** widget in a mashup:

1. In ThingWorx Composer, open the Mashup Builder and create a new mashup. Drag the **File Upload** widget into the mashup and save the mashup with a name. (This step is only necessary the first time you use the widget.)
2. Click **View Mashup** and enter the following information:
   - **File Repository** – Select the **AnaltyicsUploadStorage** as the target repository for the uploaded file.
   - **Path on Repository** – Enter a file path inside the **AnalyticsUploadStorage** repository where you want to store the uploaded file (to store at the root level, just enter /).
   - **Choose File** – Click and navigate to select the file you want to upload..



3. Click **Upload**. The selected file is uploaded to **AnalyticsUploadStorage** and can be used to create a dataset.

Note

In addition to uploading data, the File Upload widget can also be used for uploading PMML models to the **AnalyticsUploadStorage** repository. From the repository, these models can then be uploaded to the Results microservice (using the **Upload Model** service) where they can be accessed for use during other jobs by providing the **modelUri** (such as: `results:/models/<jobId>`).

Upload via REST Client

To upload data and metadata via a REST client, use the ThingWorx Core **FileRepositoryThing** API class. Adapt the **CreateFileText** method to upload data and metadata to the **AnalyticsUploadStorage** Thing.

For information about communicating with ThingWorx Core REST endpoints, see ThingWorx REST API in the ThingWorx Core Help Center.

Upload via Analytics Builder

Upload data and metadata via Analytics Builder:

1. If the ThingWorx Analytics Extension has been installed and configured, open a browser and log into Analytics Builder. (For more information about installation and configuration, see Getting Started with Analytics Builder.)
2. Follow the instructions to Create and Configure a New Dataset.

Note

Because ThingWorx Analytics functionality is now native to ThingWorx Core, the Upload Thing is no longer necessary in order to upload data using Analytics Builder.

Upload Large Volume Data Directly

For large volume data, it can be useful to bypass other file upload methods and transfer the data directly to a file storage repository. If you have file system access on the server where an instance of ThingWorx Core is installed, you can use a file transfer tool (such as an ftp application) to load the data directly into the **AnalyticsUploadStorage** file repository at the following path:

```
/ThingworxStorage/repository/AnalyticsUploadStorage
```

7

Create a Dataset

Before creating a dataset, make sure you have completed the following steps:

- Prepare the data and metadata files.
- Upload the data and metadata files to the AnalyticsUploadStorage Thing in ThingWorx Core.

The process for creating and optimizing a dataset has been streamlined. You no longer need to create the dataset and then optimize it in a separate job. One service on the Data Microservice Thing handles both of these activities.

1. In ThingWorx Composer, navigate to the **Things** page.
2. Find the **DataThing** in the list of Things and click to open it. The **General Information** page is displayed.

> **Note**
>
> The **DataThing** name will be prefixed with the server name you assigned during the ThingWorx Analytics Server installation process.

3. Using the drop down menu at the top of the screen, navigate to the **Services** page.
4. In the **Inherited Services** table, click **CreateDataset** to open the service.

5. Supply the following inputs:

Input Field	Description
csvURI	Enter the location and name of a CSV data file that was uploaded to the **AnalyticsUploadStorage** Thing. For information about uploading data and metadata files, see Upload the Data. Example: `thingworx://AnalyticsUploadStorage/<file path/name>.csv`
csvHasHeaders	If the CSV data file includes a header row, select **True**. Otherwise, select **False**.
metadataFileURI	Enter the location and name of the JSON metadata file that was uploaded to the **AnalyticsUploadStorage** Thing. For information about uploading data and metadata files, see Upload the Data. `thingworx://AnalyticsUploadStorage/<file path/name>.json`

Tip

Save Input Set – If you plan to re-use the input information for a given service, you can use a saved input set as a shortcut to populate the input fields. To use this feature, click **Save Input Set**, enter a name for the saved set of values, and click **Save Input Set**. The next time you need to execute the service with the same inputs, click the arrow in the **Select input set** field at the top of the **Inputs** panel and select the saved input set.

6. Click **Execute**. The new dataset is created and a Job ID for the results is returned.
7. To obtain the dataset URI for use in other services:
   - Copy the **jobId**.
   - Close the **CreateDataset** service page.
   - Navigate to the **GetJobInfo** service and click to open it.

- Paste the copied ID into the **jobId** field and click **Execute**. Job information details are displayed in the **Output** panel.
- Scroll across to the **datasetRef** column and click the infotable icon. The **datasetUri** and **format** are displayed in the Output panel.

Tip

Steps to find a list of the files that have been uploaded to the **AnalyticsUploadStorage** Thing:

- On the Things page, find and open the **AnalyticsUploadStorage** Thing. If it is not listed, click the **Filter Options** icon at the top of the page and select **Show System Objects**.
- When the **AnalyticsUploadStorage General Information** page opens, use the drop down field at the top to navigate to the **Services** page.
- In the **Inherited Services** table find and open the **GetFileListingWithLinks** service.
- In the **Inputs** panel, click **Execute**. A list of the files currently available in the **AnalyticsUploadStorage** Thing will be listed in the **Output** panel.

8

Retrieve a List of Connected Microservices

Each Analytics microservice is represented in ThingWorx as a Thing. Each microservice Thing is based on a Thing Template from which it inherits the available Thing Services. For an overview of how Things, Thing Templates, and Thing Services work together, see the ThingWorx Core Help Center.

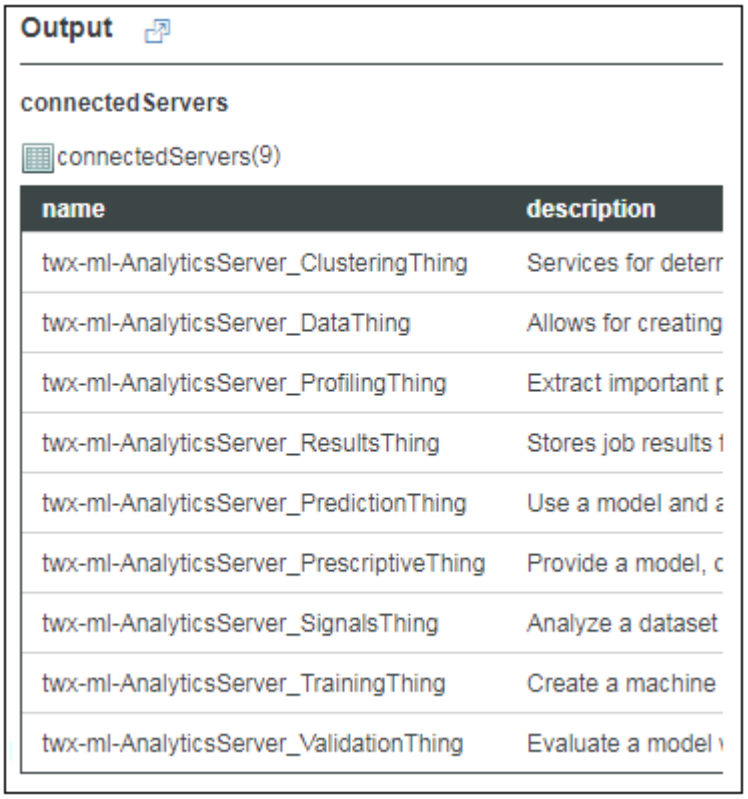
When you first start working with the Analytics microservices, you might want to retrieve a list of all the connected microservice Things available:

1. Open ThingWorx Composer and navigate to the Thing page.
2. Locate the **AnalyticsServer** in the list of Things and click to open it. The **General Information** page is displayed.

> **Note**
>
> The **AnalyticsServer** name will be prefixed with the server name you assigned during the ThingWorx Analytics Server installation process.

3. Using the drop down menu at the top of the screen, navigate to the **Services** page.
4. In the **Inherited Services** table, find the **RetrieveAnalyticsServers** service and click to open it. The service page opens.
5. In the **Inputs** panel of the service page, click the **Execute** button to run the service.

The service retrieves a list of the connected Analytics microservices. The list includes each microservice Thing name, a description, and the name of its baseThing Template.



9

Test Microservice Functionality

Before building ThingWorx Analytics microservice functionality into a ThingWorx mashup or your own code environment, you might want to explore the inputs and outputs of specific services. To interact with the Analytics microservices, for test purposes:

1. Log into ThingWorx Composer.
2. Browse the list of Things and select an Analytics microservice Thing to open it.
3. Navigate to the **Services** page.
4. Locate the service you want in the **Inherited Services** table and open it.
5. Enter any required input information.
6. Execute the service.

10

Include Data in a Request Body

In some situations, you might want to bypass the data upload and dataset creation procedures and access in-place data directly from a CreateJob request. The data and metadata must be included in the form of infotables that are passed as input parameters when the request is submitted.

Note

This technique works best for small amounts of data, so as not to strain the system constraints.

There are multiple methods available to include data in the body of a CreateJob request:

- In a REST API request
- In a ThingWorx Composer mashup (other than Analytics Builder)
- In a ThingWorx Composer javascript service

Regardless of the request method you use, you must first prepare both the AnalyticsDatasetRef and the AnalyticsDatasetMetadata infotables for inclusion in the request. For more information, see Key Analytics Infotables.

The metadata infotable has a static structure. But keep the following in mind when preparing the dataset infotable:

- **datasetURI** – Must take the following form: `body:/`
- **format** – Must be: `CSV`
- **data** – Must be provided in the form of an infotable that includes only primitive base type fields (STRING, INTEGER, NUMBER).
- **metadata** – Must be provided in the form of an infotable. The structure of this infotable is defined and static (AnalyticsDatasetMetadata).

For a sample of a training request, see the Sample Post Body section.

Note

There is one difference between submitting the CreateJob request via a REST call versus in ThingWorx Composer via a mashup or a javascript service. In ThingWorx Composer, the **data** and **metadata** parameters are mapped to an infotable object. But in a REST call, a JSON version of the infotable must be built to map these parameters.

For information about communicating with ThingWorx Core REST endpoints, see ThingWorx REST API in the ThingWorx Core Help Center.

11

Key Analytics Infotables

In ThingWorx Analytics Server, two key infotables are in use for providing dataset and metadata information to a job request. Regardless of how the request is submitted (via a REST call, a mashup, or a service), the dataset and metadata must be provided in the form of an infotable.

An infotable is an instance of a data shape that includes data. Two commonly–used infotables are the following:

- AnalyticsDatasetRef – This infotable references a specific dataset, including a URI and the data format (both will vary depending on whether the data is a stored dataset or in-place data provided with a request). The **data** field in AnalyticsDatasetRef can be based on any data shape, as long as it is flat (does not contain any nested infotables) and it matches the corresponding metadata infotable.
- AnalyticsDatasetMetadata – This infotable is a data shape containing the machine learning characteristics that describe a dataset, including fieldName, dataType, opType, and more.

Note

Metadata only needs to be provided if you are accessing data that is not already in the Data microservice, that is, the data URI does not use the scheme `dataset://`.

AnalyticsDatasetRef

The **data** field in this infotable can be based on any Data Shape as long as it is flat (contains no nested infotables) and it matches the corresponding metadata. Each field of data must be defined according to the structure established in the metadata infotable. For additional information about creating and working with Data Shapes, see the Data Shapes section of the ThingWorx Core Help Center.

The AnalyticsDatasetRef contains the following fields:

- **datasetUri** – A string that points to the location of the data you want to include in the request. Data can be accessed from a variety of locations. The syntax of this field includes two components separated by a colon. On the left side of the colon is a scheme, which indicates the source of the data (such as a stored dataset or in-place data provided directly to a request). On the right side of the colon is a path, which provides the specific location. The possible schemes include:
  - `thingworx://` – Points to a ThingWorx file repository, such as AnalyticsUploadStorage, where data can be uploaded and stored.
  - `thingworxs://` – Functions the same as the `thingworx://` scheme but is used when ThingWorx Core server is accessed over SSL (https).
  - `file:/` – Points to the ThingWorx Analytics Server file system where data can be loaded directly and accessed in place. Accessing data from the file system is useful for small datasets with rapidly changing data.
  - `body:/` – Points to in-place data that can be supplied directly as part of an API request body. This method is useful for scoring or model evaluation jobs. Data supplied this way is not stored.
  - `dataset:/` – Points to a dataset created and stored in Data microservice.
- **format** – A string that indicates the storage format of the data. Supported values include:
  - `CSV` – For use with the `thingworx://`, `thingworxs://`, `body:/` or `file:/` schemes
  - `PARQUET` – For use with the `dataset:/` scheme
- **filter** – A string that contains clause conditions for an SQL WHERE statement to describe the characteristics of the data that should be included. It has the effect of removed rows of data from the dataset (does not remove columns).
- **exclusions** – A list of strings that remove specific fields from the dataset. It has the effect of removed columns from each row in the dataset (does not remove rows).
- **data** – An untyped infotable that must be provided in order to pass data as part of a request body. It can accept any infotable that includes only primitive base type fields (STRING, INTEGER, NUMBER).
- **metadata** – A string that points to the metadata infotable and must be provided in order to pass metadata as part of a request body.

AnalyticsDatasetMetadata

This infotable is a static data shape containing the machine learning characteristics that describe a dataset. It includes the following fields:

- **fieldName** – A string that provides the field name for a column of data.
- **dataType** – A string that indicates the format of the data in this field. Acceptable values include:
  - DOUBLE
  - INTEGER
  - LONG
  - STRING
  - BOOLEAN
- **opType** – A string that indicates how the data behaves. Acceptable values include:
  - CONTINUOUS – A numeric field (DOUBLE, INTEGER, or LONG dataType) that can include data of any value across the provided range. Data in this opType usually falls between the **min** and **max** values.
  - BOOLEAN – A true or false field, typically used as a flag.
  - ORDINAL – A field with a finite set of values that have an inherent ordering. Only the provided **values** are acceptable and they must be provided in the correct order.
  - CATEGORICAL – A field with a finite set of values, where each value is treated independently and there is no ordering. Only the provided **values** are acceptable unless no values are provided. In that case, **values** are calculated from all observed entries within the dataset.
  - ENTITY_ID – For time series data, this identifier allows for differentiation between entities. For example, if a dataset contains entries for three different machines, each machine should have its own entity ID.
  - TEMPORAL – For time series data, this time value field indicates the expected time between adjacent rows of data (provided in the **timeSamplingInterval**).
  - INFORMATIONAL – An informational field that is not considered for machine learning algorithms.
- **min** – For continuous values, this field represents the lowest expected value. This is an informational field.
- **max** – For continuous values, this field represents the highest expected value. This is an informational field.
- **values** – For ordinal and categorical values, this field contains a list of possible values. For ordinal, the values must be listed in the correct order. For categorical, the order of values doesn't matter.

- **timeSamplingInterval** – For time series datasets, this value indicates the time interval between adjacent rows of data. If the dataset does not adhere to the specified interval, an error will occur.
- **isStatic** – For time series datasets, this flag can be used to indicate that a field should not change over time. When set to true, this field will not undergo any time series transformations. Setting this flag where appropriate will improve performance.

12

Sample Post Body

The following sample represents the post body of a Create Job Training request pointing to a dataset on the Data Microservice. This sample includes two infotables, the datasetRef infotable (which, in turn, contains the AnalyticsDatasetMetadata infotable) and the learners infotable. For more information about the structure of the training request and the infotables, see the **API documentation** in the ThingWorx Analytics Help Center.

```
{
  "jobName": "beanPro Grinder Error model",
  "validationHoldoutPercentage": 0.2,
  "maxAllowedFields": 50,
  "goalField": "Grinder Error Occurrence",
  "description": "predicting the grinder error occurence value on the beanpro
dataset",
  "datasetRef": {
    "created": 1501687620663,
    "description": "",
    "name": "Infotable",
    "dataShape": {
      "fieldDefinitions": {
        "filter": {
          "name": "filter",
          "aspects": {},
          "description": "Refine the dataset by filtering out values",
          "baseType": "STRING",
          "ordinal": 3
        },
        "datasetUri": {
          "name": "datasetUri",
          "aspects": {},
          "description": "Location of the data",
```

```
        "baseType": "STRING",
        "ordinal": 1
      },
      "metadata": {
        "name": "metadata",
        "aspects": {
          "dataShape": "AnalyticsDatasetMetadata"
        },
        "description": "Metadata describing the locally provided data",
        "baseType": "INFOTABLE",
        "ordinal": 6
      },
      "data": {
        "name": "data",
        "aspects": {},
        "description": "Provide the data locally as part of the request",
        "baseType": "STRING",
        "ordinal": 5
      },
      "format": {
        "name": "format",
        "aspects": {},
        "description": "Structure of the data (eg: csv, json)",
        "baseType": "STRING",
        "ordinal": 2
      },
      "exclusions": {
        "name": "exclusions",
        "aspects": {
          "dataShape": "GenericStringList"
        },
        "description": "Refine the dataset by removing fields",
        "baseType": "INFOTABLE",
        "ordinal": 4
      }
    },
    "name": "AnalyticsDatasetRef",
    "description": "Pointer to an analytics dataset"
  },
  "rows": [
    {
      "datasetUri": "dataset:///c2b17f4b-6273-4e04-a466-26f0755e8af3",
      "metadata": {
        "created": 1501687665676,
        "description": "",
        "name": "Infotable",
```

```
          "dataShape": {
            "fieldDefinitions": {
              "timeSamplingInterval": {
                "name": "timeSamplingInterval",
                "aspects": {},
                "description": "Time in between observations in Temporal field",
                "baseType": "INTEGER",
                "ordinal": 7
              },
              "isStatic": {
                "name": "isStatic",
                "aspects": {},
                "description": "If this field remains unchanged over time and
shouldn't be pivoted for time series transformations",
                "baseType": "BOOLEAN",
                "ordinal": 8
              },
              "fieldName": {
                "name": "fieldName",
                "aspects": {},
                "description": "Name of the field",
                "baseType": "STRING",
                "ordinal": 1
              },
              "min": {
                "name": "min",
                "aspects": {},
                "description": "Minimum observed value for Continuous fields",
                "baseType": "NUMBER",
                "ordinal": 4
              },
              "max": {
                "name": "max",
                "aspects": {},
                "description": "Maximum observed value for Continuous fields",
                "baseType": "NUMBER",
                "ordinal": 5
              },
              "dataType": {
                "name": "dataType",
                "aspects": {},
                "description": "Format of the data (STRING,DOUBLE,BOOLEAN,INTEGER)",
                "baseType": "STRING",
                "ordinal": 2
              },
              "opType": {
```

```
                "name": "opType",
                "aspects": {},
                "description": "Behavior of the data (CONTINUOUS,CATEGORICAL,ORDINAL,
BOOLEAN,TEMPORAL,ENTITY_ID)",
                "baseType": "STRING",
                "ordinal": 3
              },
              "values": {
                "name": "values",
                "aspects": {
                  "dataShape": "GenericStringList"
                },
                "description": "Collection of possible values for Ordinal and
Categorical fields",
                "baseType": "INFOTABLE",
                "ordinal": 6
              }
            },
            "name": "AnalyticsDatasetMetadata",
            "description": "Metadata describing a dataset field"
          },
          "rows": []
        },
        "format": "parquet",
        "exclusions": {
          "created": 1501687665679,
          "description": "",
          "name": "Infotable",
          "dataShape": {
            "fieldDefinitions": {
              "item": {
                "name": "item",
                "aspects": {
                  "isPrimaryKey": true
                },
                "description": "Item",
                "baseType": "STRING",
                "ordinal": 0
              }
            },
            "name": "GenericStringList",
            "description": "Generic data shape to hold a list of strings"
          },
          "rows": []
        }
      }
```

```
    ]
  },
  "learners": {
    "created": 1501687620656,
    "description": "",
    "name": "Infotable",
    "dataShape": {
      "fieldDefinitions": {
        "maxDepth": {
          "name": "maxDepth",
          "aspects": {},
          "description": "The maximum tree depth (DECISION_TREE, RANDOM_FOREST,
and GRADIENT_BOOST only)",
          "baseType": "INTEGER",
          "ordinal": 2
        },
        "learningTechnique": {
          "name": "learningTechnique",
          "aspects": {},
          "description": "The technique to use for learning (NEURAL_NET,
LINEAR_REGRESSION, LOGISTIC_REGRESSION, DECISION_TREE,
RANDOM_FOREST, and GRADIENT_BOOST)",
          "baseType": "STRING",
          "ordinal": 1
        },
        "hiddenUnitPercentage": {
          "name": "hiddenUnitPercentage",
          "aspects": {},
          "description": "The percentage of the number of input nodes to use
in each hidden layer of a NEURAL_NET",
          "baseType": "NUMBER",
          "ordinal": 5
        },
        "layerCount": {
          "name": "layerCount",
          "aspects": {},
          "description": "The number of layers to use in NEURAL_NET",
          "baseType": "INTEGER",
          "ordinal": 4
        },
        "numberOfIterations": {
          "name": "numberOfIterations",
          "aspects": {},
          "description": "The number of iterations for GRADIENT_BOOST",
          "baseType": "INTEGER",
          "ordinal": 6
```

```
        },
        "treeCount": {
          "name": "treeCount",
          "aspects": {},
          "description": "The number of trees to use in RANDOM_FOREST",
          "baseType": "INTEGER",
          "ordinal": 3
        },
        "removeDuplicatesAndUniformColumns": {
          "name": "removeDuplicatesAndUniformColumns",
          "aspects": {},
          "description": "Determines whether to remove duplicate columns for
DECISION_TREE",
          "baseType": "BOOLEAN",
          "ordinal": 6
        }
      },
      "name": "AnalyticsTrainingLearner",
      "description": "The configuration of a machine learning model for a
 training job"
    },
    "rows": [
      {
        "learningTechnique": "NEURAL_NET"
      }
    ]
  }
}
```

13

Sample REST API Call

In this release, all REST calls are handled via the ThingWorx Core REST layer. The following sample represents the syntax required to run a Create Job Training request. Replace the content in the angled brackets with specific information for your environment.

```
http://<ThingWorx Core Host>:<ThingWorx Core Port>/
Thingworx/Things/<AnalyticsServer name>_TrainingThing/
Services/CreateJob
```

For more information about communicating with ThingWorx Core REST endpoints, see ThingWorx REST API in the ThingWorx Core Help Center.

14

Mapping Services to Legacy REST APIs

AnalyticsGateway

Service	REST Endpoint Replaced	Notes
RetrieveAnalytics-Servers	new	Returns a list of the connected microservices.

Analytics Server

This server is not directly accessible, but its service is inherited and available for use in each of the subsequent microservice Things.

Service	REST Endpoint Replaced	Notes
VersionInfo	Retrieve Version Information	Returns the internal version number for a specific microservice. The first four digits = ThingWorx Core version. The next six digits = version of the microservice.

AnalyticsJob Server

This server is not directly accessible, but its services are inherited and available for use in each of the subsequent microservice Things.

Service	REST Endpoint Replaced	Notes
AddJobTags	new	Adds tags to a job.
DeleteJob	new	Deletes a job that is finished, whether it failed or not.
DeleteJobTags	new	Deletes tags from a job.
GetJobInfo	Retrieve a Specific Signals Job Retrieve a Specific Clusters Job Retrieve Specific Profiles Job Retrieve Specific Prediction Job Retrieve Specific Predictive Scoring Job Retrieve Specific Predictive Evaluation Job	Retrieves information about a specific job.
GetJobInfos	List Signals Jobs List All Clusters Jobs List All Profiles Jobs List All Prediction Jobs List All Predictive Scoring Jobs List All Predictive Evaluation Jobs	Retrieves a list of all jobs for a given microservice.
GetJobStatus	Retrieve Job Status Retrieve the Status of a Specific Job	Retrieves the status of a specific job.
GetJobStatuses	Retrieve All Pending Jobs	Retrieves a list of job statuses, for a given microservice, in chronological order (with pending jobs appearing first).

Service	REST Endpoint Replaced	Notes
GetJobTags	new	Lists the available job tags.
GetPage	new	Moves to next/previous page.
ReplaceJobTags	new	Replaces tags on a job.
SaveJobResult	new	Save results as a file and stores it in the **AnaltyicsResultsStorage** repository. Can store PMML, CSV, or JSON. A URL to the results is returned.

Results Microservice

Service	REST Endpoint Replaced	Notes
AddResultsTags	new	Adds tags to the results of a job.
DeleteResult	Delete Signals Job Delete Clusters Job Delete Profiles Job Delete Prediction Job Delete Predictive Scoring Job Delete Prescriptive Scoring Job Delete Predictive Evaluation Job	Deletes results of a specific job, without deleting the job itself.
DeleteResultTags	new	Deletes tags from results.
GetDetails	new	Returns job details, such as date and time, queued time, duration.
GetDetailsList	new	Returns a page of details

Service	REST Endpoint Replaced	Notes
		for all results by result type.
GetPage	new	Moves to next/previous page.
GetResultTypes	new	Returns a list of supported result types.
GetResultsTags	new	Returns a list of results tags.
QueryInputFields	new	Queries the PMML model (for training or clustering) and returns the fields required as input for scoring or evaluation.
QueryOutputFields	new	Queries the PMML model for output fields expected from scoring or evaluation.
ReplaceResultsTags	new	Replaces tags on results.
SaveResult	new	Save results as a file and stores it in the **AnaltyicsResultsStorage** repository. Can store PMML, CSV, or JSON. A URL to the results is returned.
UpdateNameAndDescription	new	Allows changes to the information returned by GetDetails.
UploadModel	new	Allows upload of an existing PMML model from the **AnalyticsUploadStorage** file repository to the Results microservice.

Data Microservice

Service	REST Endpoint Replaced	Notes
AppendDataset	new	Adds additional data to an existing dataset from a CSV file.
BinnedDistributionQuery	Binned Distribution Query	An asynchronous (batch) job that queries the record count for specified bins of a continuous field in a dataset.
BinnedDistributionQuery-Synchronous	new	Runs the BinnedDistribution-Query service synchronously so that results are returned immediately on job completion.
CreateDataset	Create Dataset	Creates the dataset and optimizes it automatically.
DistributionQuery	Query Distribution of Data	Queries the record count for specified fields in a dataset (non-continuous fields only).
DistibutionQuerySyn-chronous	new	Runs the DistributionQuery service synchronously so that results are returned immediately on job completion.
DistributionStatistics-Query	new	Queries validation statistics separately from retrieving predictive model results.
DistributionStatistics-QuerySynchronuous	new	Runs the DistributionStatistics-Query service

Service	REST Endpoint Replaced	Notes
		synchronously so that results are returned immediately on job completion.
GetDatasetSchema	RetrieveDataset Configuration	Retrieves the metadata from a dataset.
ListCreatedDatasets	List Datasets	Retrieves a list of created datasets.
RetrieveBinnedDistribu-tionQuery	new	An asynchronous (batch) job that returns BinnedDistribution-Query results.
RetrieveDistribution-Query	new	An asynchronous (batch) job that returns DistributionQuery results.
RetrieveDistributionSta-tistics	new	An asynchronous (batch) job that returns DistributionStatistics-Query results.

Profiling Microservice

Service	REST Endpoint Replaced	Notes
CreateJob	Submit Profiles Job	Creates a job to generate profiles.
RetrieveResult	Retrieve Profiles Result	Retrieve the results of a profiles job.

Signals Microservice

Service	REST Endpoint Replaced	Notes
CreateJob	Submit Signals Job	Creates a job to identify signals.
RetrieveResult	Retrieve Signals Result	Retrieve the results of a signals job.

Clustering Microservice

Service	REST Endpoint Replaced	Notes
CreateJob	Submit Clusters Job	Create a clusters model.
RetrieveModel	Retrieve Clusters Results	Retrieves a clustering model

Training Microservice

Service	REST Endpoint Replaced	Notes
CreateJob	Submit Prediction Model Generation Job	Create a prediction model job.
RetrieveModel	Retrieve and Describe Prediction Model Result	Only retrieves the PMML model. But if a holdout for validation was specified in the CreateJob, a validation job is auto-created and runs.

Prediction Microservice

Service	REST Endpoint Replaced	Notes
BatchScore	Submit Predictive Scoring Job	Run prediction scoring in an asynchronous process. Results are persisted and can be accessed for subsequent use.
RealTimeScore	new	Run synchronous predictive scoring. Results are not persisted.
RetrieveResult	Retrieve Predictive Scoring Result and Retrieve Predictive Scoring Important Feature Result	Retrieves results from prediction scoring jobs.

Prescriptive Microservice

Service	REST Endpoint Replaced	Notes
RealTimeScore	new	Run synchronous prescriptive scoring (for small datasets).

Asynchronous (batch) prescriptive scoring is not currently available in the 8.1 release. It will be reintroduced in a future release.

Validation Microservice

Service	REST Endpoint Replaced	Notes
CreateJob	Submit Predictive Evaluation Job	Run a job to validate the efficacy of a specified model with a specified dataset.
RetrievePVAS	Query Predicted Results vs. Actual Prediction Results	Retrieves the PVAs of a validation job.
RetrieveResult	Retrieve Predictive Evaluation Result and Query Predicted Results vs. Actual Prediction Results	Output will vary based on the `optype` of the goal variable.

15

Frequently Asked Questions

Can I continue to use models that were generated in a previous version of ThingWorx Analytics?

Yes, legacy models can be imported. In addition, external models created in other systems can also be imported, as long as they conform to the PMML 4.3 standard.

To use a legacy model, first export it from the previous version of ThingWorx Analytics as follows:

1. Open your REST client, select the **GET** method, and enter the following URI:

   `https://<IP address>/1.0/datasets/<dataset name>/prediction/<result ID>/export.pmml`

   Where:

   - **dataset name** is the name of the legacy dataset you want to export
   - **result ID** is the ID of the predictive model generation job that created the legacy model
2. Set the **Content-Type** and **Accept** headers to `text/xml`.
3. Submit the request. The model will be exported and can be saved for further use.

To import legacy or external PMML models into ThingWorx Analytics 8.1:

1. Upload the model to the AnalyticsUploadStorage file repository, using either the **File Upload** widget or another mashup.
2. Open the ResultsThing and navigate to the Services page.
3. Use the **Upload Model** service to upload the stored PMML model from the AnalyticsUploadStorage file repository to the ResultsThing where it can be accessed for use during other jobs by providing the **modelUri** (such as: `results:/models/<jobId>`)..

Can I continue to use REST jobs that were created in a previous version of ThingWorx Analytics?

No, the legacy REST API calls will need to be rewritten for two reasons. First, all support for REST calls will be handled by a ThingWorx Core REST layer, which will require a new URL pattern for access (see Sample REST API Call). Second, datasets are no longer included in the REST URL. Instead, datasets and metadata are now referenced via URI information supplied in infotables, However, in this release, the integration with ThingWorx Core also makes it possible to access ThingWorx Analytics functionality via ThingWorx Composer mashups.

Can I continue to use my existing datasets?

No, existing datasets cannot be used as they are. Both the way data is stored and the structure of the metadata have changed. However, you can reload the original, raw data and use it to create new datasets. For more information, see Prepare Data and Metadata.

How will my output results compare to previous results?

Model output will still be in the form of PMML output, although with an upgrade to the PMML 4.3 standard. In terms of scoring results, if you run scoring against older models, your scoring results should be the same as with previous versions of the software. However, if you score against models built in the new microservice architecture, you will find improved model performance.

Benchmark testing against the previous release shows the following about ThingWorx Analytics 8.1:

- Signals – Performance and quality are both equivalent to the previous release.
- Profiles – There are some improvements in profile quality.
- Clustering – This process has been updated to true, unsupervised clustering that is no longer goal-based. The new process takes a bit longer but provides better quality clusters.
- Training – This functionality has been updated to process the entire dataset (except for a validation holdout) rather than performing iterative sampling. The new process takes a bit longer but produces more accurate models.
- Predictive scoring – This process is much faster due to the streamlining in the dataset preparation process. In addition, for small sets of data, the upload process can be bypassed and data can be used directly in the service calls.
- Prescriptive scoring – Synchronous scoring is faster. (Batch scoring is not available in this release.)
- Use of infotables – The use of infotables to input and output data is new structurally, but does not affect processing performance.

How do I ensure that the ThingWorx Analytics Server and the ThingWorx Core communicate properly?

The interaction between the two servers is facilitated behind the scenes by an Edge Agent application. For it to do its job properly, you need to ensure the following during ThingWorx Analytics Server installation:

- Install the ThingWorx Core first and make sure it's running.
- Generate an application key in ThingWorx Core.
- Make sure there is network connectivity between ThingWorx Core and ThingWorx Analytics.
- Provide connection information, including: IP Address, Port, and a Thing name for the Analytics Server. For more information, see one of the *Installation Guides* (Docker or Native Linux) in the Reference Documents section of the PTC eSupport Portal.

Once the installation is complete, you can verify the connection status as follows:

1. In ThingWorx Composer, browse the list of Things to find the AnalyticsServer Thing that was created during installation (it will be prefixed by the name you assigned during installation).
2. Open the AnalyticsServer Thing and navigate to the Properties and Alerts page.
3. In the **AnalyticsGateway** properties table, find the **isConnected** property and make sure the check box in the **Value** column is checked.

You can also generate a list of connected microservices to ensure everything is connected and running properly. For information about this process, see Retrieve a List of Connected Microservices.