IBM DECISION OPTIMIZATION CONNECTOR FOR IBM® Cognos® TM1® AND IBM® PLANNING ANALYTICS

v. 1.1.0
July 27, 2016

IBM Analytic Solutions

by

IBM
Introduction

Optimization can be used to develop prescriptive solutions and scenarios for many different business problems, including production planning and scheduling.

The “IBM Decision Optimization Connector for IBM Cognos TM1 and IBM Planning Analytics” is designed to execute optimization scenarios using IBM Decision Optimization on Cloud (DOcloud) from data held in either IBM Cognos TM1 or in IBM Planning Analytics. For simplicity in this document, we shall collectively use

- \{TM1/PA\} for TM1 and for Planning Analytics
- \{DO\} for IBM CPLEX® Optimization Studio and for DOcloud

This Connector transmits data from \{TM1/PA\} to DOcloud, and returns the optimized results to \{TM1/PA\}. Setup is required, and described in detail throughout this documentation.

Product Requirements

- Planning and Forecasting using \{TM1/PA\}, either:
  - IBM Cognos TM1 (10.2.2.1 or greater)
    - IBM Cognos Configuration
    - IBM Cognos TM1 Server
    - IBM Cognos TM1 Architect or IBM Cognos TM1 Performance Modeler
  OR
  - IBM Planning Analytics (10.3 or greater)
    - IBM Cognos Configuration
    - IBM Cognos TM1 Server
    - IBM Cognos TM1 Architect or IBM Cognos TM1 Performance Modeler
- Decision Optimization (DO) Products and Services:
  - For Model building: IBM CPLEX Optimization Studio (12.2 or greater)
  - For Execution: IBM Decision Optimization on Cloud [DOcloud]

Initial Considerations

We have assumed that you have already built or will build matching models in \{TM1/PA\} and \{DO\}. This documentation does not cover model building in \{TM1/PA\} nor in \{DO\}. This documentation does help experienced users to map the variables, metadata, and data to enable the Connector to move data between \{TM1/PA\} and DOcloud, to use information in a \{TM1/PA\} model as the data source for a \{DO\} model, to execute the \{DO\} model and to store the optimized results in the \{TM1/PA\} model.
Preliminary Steps

{TM1/PA} Server Configuration

1) Ensure the {TM1/PA} Model is set up for the TM1 REST API. Most importantly, specify the HTTPPortNumber parameter to the {TM1/PA} server configuration file, as in Figure 0.1. Depending on your situation, this (and restarting the {TM1/PA} server) may be the only step required. Please review the above link to verify.

```
[TM15]
    ## REST API parameters
    HTTPPortNumber=8111
```

Figure 0.1: Example of additional {TM1/PA} REST API parameter in tm1s.cfg file.

Note: The current version of the DO-TM1 Connector only supports {TM1/PA} servers configured for IntegratedSecurityMode = 1, 4, or 5.

2) Refer to the Java Extensions Support section in the TM1 TurboIntegrator Guide to enable and configure Java Extensions for your TM1 server. Go through each of the following steps:

   a) Configure the TM1 server to support Java. One possible path comes with the standard {TM1/PA} installation: `<TM1 install path>\bin64\jre\7.0\bin\default\jvm.dll`

   b) Create or edit the javaextensions.policy file.

   c) The DO-TM1 Connector does not require you to create the Java extensions directories. The required directory for the TM1 server is included in the Connector package, and is added in section 2.

{TM1/PA} Model

Figure 0.2: Cubes and dimensions of the example {TM1/PA} business model discussed in this document.
Figure 0.3: \{DO\} example gas model.
**{DO} Model**

A good OPL reference is the [IBM ILOG OPL Language Reference Manual](https://www.ibm.com). The Connector supports a single {DO} model (.mod) file when running an on-premises optimization, and either a single model file or an OPL project with multiple model files for DOcloud.

A revised example “gas” model is shown in Figure 0.3. We have made minor changes to accommodate the Connector, such as changing all tuple fields to basic data types (int, float, string), the corresponding references to those tuple fields, and the output variable definition in the postprocessing block. The original file is located in the path `<CPLEX Optimization Studio install path>/opl/examples/opl/gas/gas.mod`

See section 9 for more guidelines on how to build a {DO} model to bring in data from {TM1/PA}.

### 1. Example Business Problem

Based on the {DO} example model in Figure 0.3 above.

Three components (nitrogen, hydrogen, and chlorine) are used to make two products (gas, chloride).

Producing one unit of gas requires (demands) 1 unit of nitrogen, and 3 units of hydrogen. It nets $30 in profit per unit sold.

Producing one unit of chloride requires (demands) 1 unit of nitrogen, 4 units of hydrogen, and 1 unit of chlorine. It nets $40 in profit per unit sold.

With current stock levels of: 50 units of nitrogen, 180 units of hydrogen, and 40 units of chlorine, how many units of each product should be made to earn the highest profit?
2. Downloading the Connector

Figure 2.1: Folder structure of the bundled Connector zip file.

1) Stop the {TM1/PA} server.

2) Copy the javaextensions folder into the {TM1/PA} server data files directory. Note: if your {TM1/PA} server already contains the javaextensions folder, merge the copied folder with your own or copy the files and folders individually into your existing javaextension folder.

3) Copy all of the files from the TM1_Objects folder into the location of the {TM1/PA} server data files.

4) Add the following to your {TM1/PA} server’s tm1s.cfg file:

```
JavaClassPath=..\javaextensions\libraries\jackson-annotations-2.5.1.jar;..\javaextensions\libraries\jackson-core-2.5.1.jar;..\javaextensions\libraries\jackson-databind-2.5.1.jar;..\javaextensions\libraries\api_java_client-1.0-STABLE-b36.jar;..\javaextensions\libraries\commons-codec-commons-codec-1.9.jar;..\javaextensions\libraries\commons-lang3-3.3.2.jar;..\javaextensions\libraries\commons-logging-1.1.2.jar;..\javaextensions\libraries\httpclient-4.3.6.jar;..\javaextensions\libraries\httpclient-cache-4.3.6.jar;..\javaextensions\libraries\httpcore-4.3.3.jar;..\javaextensions\libraries\icu4j-53_1.jar;..\javaextensions\libraries\junit-4.10.jar;..\javaextensions\libraries\log4j-1.2.16.jar;..\javaextensions\libraries\mockito-all-1.9.5.jar
```

5) Restart the {TM1/PA} server.
3. Configuring the Connector

Open the connector.properties file in a text editor. The configuration parameters are case-sensitive:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tm1.adminhost</td>
<td>This is specified in the tm1s.cfg file of the {TM1/PA} server as the Admin Host parameter. If blank, then this parameter equals localhost. The Connector only supports one Admin Host.</td>
</tr>
<tr>
<td>tm1.adminPort</td>
<td>TM1 Admin Server port number for the REST API. The default value is 5895, unless changed per step 2 of the REST API instructions.</td>
</tr>
<tr>
<td>tm1.adminSSL</td>
<td>Does the TM1 Admin Server use SSL? (T or F). The default value is F.</td>
</tr>
<tr>
<td>api.url</td>
<td>URL for the DOcloud API. Assigned with a DOcloud subscription along with the API key.</td>
</tr>
<tr>
<td>cam.namespace</td>
<td>CAM Namespace for {TM1/PA} servers using IntegratedSecurityMode = 4 or 5.</td>
</tr>
</tbody>
</table>

Example:

```
tm1.adminhost=localhost
tm1.adminPort=5895
tm1.adminSSL=F
api.url=https://api-oaas.docloud.ibmcloud.com/job_manager/rest/v1/
cam.namespace=saml_namespace
```

4. Setting {TM1/PA} server information

The Connector requires information about the {TM1/PA} server. This is set in the Connector TM1 Server Information cube. The view is shown in Figure 4.1.
The parameters are defined as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Name</td>
<td>Name of the {TM1/PA} server.</td>
</tr>
<tr>
<td>TM1 User Name</td>
<td>You are recommended to create a {TM1/PA} user name that must minimally provide access to all necessary data to be passed to DOcloud. The associated password is encrypted in section 5.</td>
</tr>
<tr>
<td>TM1 Credentials File Path</td>
<td>File path for saving the encrypted TM1 password.</td>
</tr>
<tr>
<td>DO Credentials File Path</td>
<td>File path for saving the encrypted DOcloud API key.</td>
</tr>
<tr>
<td>Key File Path</td>
<td>File path for saving the key for decryption.</td>
</tr>
</tbody>
</table>

5. Encrypting the password and API key

In the {TM1/PA} server, run the Encrypt Credentials TurboIntegrator (TI) process. At the start of the process, you will be prompted for these two mandatory parameters:

1) {TM1/PA} password, and
2) the DOcloud API key.

The password is associated with the {TM1/PA} user name defined in section 4. The DOcloud API key is provided with a DOcloud subscription.
Once the TI process is finished, those two parameters will be stored as encrypted information in the file paths specified in section 4. This TI process only needs to be run once, unless either of the two parameters change.

6. Mapping \{TM1/PA\} and \{DO\}

The Connector relies on you to map metadata so it knows how to transfer data between \{TM1/PA\} and DOcloud. Structures in \{TM1/PA\} (i.e. cube views and dimension subsets) need to be passed to \{DO\} structures (tuples).

One dimension and three cubes are used for mapping the \{TM1/PA\} model to a single \{DO\} model:

- **DO Objects**: Dimension containing the names of tuple variables in the \{DO\} model. See section 6.3.
- **DO-TM1 Mapping**: The cube and view name for general mapping. Specifies names of \{TM1/PA\} objects – subsets/dimensions and views/cubes – that map to the \{DO\} variables. See section 6.4.
- **View Mapping**: The cube and view name for details about each mapped \{TM1/PA\} cube view. Here you choose which dimensions in a view should be mapped to \{DO\} tuple fields, and which dimensions should have a single element selected (sliced on). See section 6.5.
- **Subset Mapping**: The cube and view name for details about each mapped \{TM1/PA\} dimension subset. Here you set which dimension attributes should be mapped to \{DO\} tuple fields. See section 6.6.

6.1. How to Map \{TM1/PA\} structures to \{DO\} tuples

In Figure 6.1, the left image shows a \{DO\} tuple structure (lines 20 – 24). This structure can be applied to any number of variables, in this case it is applied to the input variable *demands* (line 44). It contains three fields: *product, component*, and *demand*. The first two fields are “keys,” which are the same conceptually as primary keys in a database table or dimension members in a TM1 cube. Tuples in this context can be considered as flattened cube views.

```plaintext
20  tuple TDemand {  
21    key string product;  
22    key string component;  
23    float demand;  
24  };  
25  {TDemand} demands = ...;
```

![Figure 6.1: \{DO\} tuple structure and an example variable (left), corresponding cube view (right)](image)

Figure 6.1: \{DO\} tuple structure and an example variable (left), corresponding cube view (right)
Table 6.1: Cube view data from Figure 6.1 in same format as \{DO\} tuple data.

<table>
<thead>
<tr>
<th>{TM1/PA}:</th>
<th>Products (dimension)</th>
<th>Components (dimension)</th>
<th>demands (view data)</th>
</tr>
</thead>
<tbody>
<tr>
<td>gas</td>
<td>nitrogen</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>gas</td>
<td>hydrogen</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>gas</td>
<td>chlorine</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>chloride</td>
<td>nitrogen</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>chloride</td>
<td>hydrogen</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>chloride</td>
<td>chlorine</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

| \{DO\}:       | product (tuple key field) | component (tuple key field) | demand (tuple non-key field) |

The mapping cubes described in the next section are used to record this information in Figure 6.2: which dimensions are mapped to which fields, and for unmapped dimensions which elements should be sliced on (i.e. selected). For subsets within \{TM1/PA\}, attributes are mapped to the tuple fields in \{DO\}.

6.2. Role of Mapped \{TM1/PA\} Objects in \{DO\}

In \{TM1/PA\}, cube views format data for analysis and planning, and for data movement. Dimension subsets in \{TM1/PA\} provide the domain (or dimensionality) for the data. It is the same concept in \{DO\} models that are designed to map to \{TM1/PA\}. The domains defined by the dimension subsets must be applied to all cube views that contain that dimension. The following examples may help clarify this concept.
Example #1:

I am a part of a company that manufactures 100 different products in 50 different production plants, and I have included optimization in this process. Both the Products and Plants dimensions in {TM1/PA} have subsets that are mapped to {DO}, and they contain the 100 products and 50 plants. Some of the data from cube views in my {TM1/PA} model that are mapped to {DO} are the demand (number of units of each product my company expects to sell) for each week. Also, the capacity of each plant in number of units of each product. Typically, I want to optimize over all 100 products and all 50 plants, since in some ways they are all interdependent.

However, I’ve received a request from a regional manager to run the optimization for only the 5 plants that he manages. This means I need to reduce my subset of plants down from 50 to 5. But that also means I need to make the same changes in any views that contain the Plants dimension. Since plants are not a part of the demand, I don’t need to change it there, but it is a part of the capacity. Within that cube view I need to change the plants to the 5 specified, or else the Connector will give an error if I leave it with all 50 included, because I’m including plants that are outside the domain of 5 plants provided by the subset.

Example #2:

Consider a time dimension with 52 weeks, W01 through W52. An optimization problem is being run for only the first 10 weeks of a planning period: W01 through W10. Those 10 elements are being passed through a subset to {DO} as the time domain for the optimization. One of the views being created has the Time dimension mapped to {DO}. Within this view, the Time dimension cannot contain any elements outside of the W01 through W10 range, or the Connector will give an error. The subset itself must be consistent with the cube views that also contain that dimension. The cube view does not need to have all those elements included, but any that are included must come from that subset.

---

**Mapping Cubes**

**Do NOT** move any dimensions in the mapping cube views that are being read by the Connector (i.e. the views defined in the DO-TM1 Mapping Management cube, the DO-TM1 Control cube and the DO-TM1 Mapping Management cube). Elements can be added in any of the dimensions, but not removed or hidden. The DO Objects dimension is a special case where only the {DO} variables should be added as elements.

---

The current version of the Connector only supports putting optimized data that was output from {DO} into {TM1/PA} cube views, not dimension subsets.

### 6.3. DO Objects dimension

The DO Objects dimension is a list of the {DO} input and output variables. These variables must be put in the same order as they appear in the {DO} model. The user must populate this dimension based on their own {DO} model.

As an example, with the {DO} model in Figure 0.3, the dimension would have the following elements:
products
components
demands
profits
stocks
productionOut

The \{TM1/PA\} element names \textbf{MUST} match the \{DO\} variable names. Since \{DO\} is case-sensitive, these elements must match the case as well.

6.4. DO-TM1 Mapping cube

The cube (and view) name is \textbf{DO-TM1 Mapping} and it contains two dimensions: \textit{DO Objects} and \textit{Mapping Parameters}. The first dimension, \textit{DO Objects}, is discussed in section 6.3.

Rules can be added to the \textbf{DO-TM1 Mapping} cube shown in Figure 6.3 to simplify data entry in this cube. For example, automatically making the TM1 Name equal to the DO Objects element name.

![Figure 6.3: Cube to define the mapping between \{TM1/PA\} and \{DO\}. \{DO\} variable names are on the Rows, mapping parameters on the columns. All \{DO\} objects are from the gas example (Figure 0.3).](image)

Table 6.2: General DO-TM1 Mapping parameters.

<table>
<thead>
<tr>
<th>DO Input/Output</th>
<th>Is it an input or output variable in DO? Currently only Views are supported as Output.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM1 Type</td>
<td>Type of {TM1/PA} object that the {DO} variable maps to (i.e. View or Subset).</td>
</tr>
<tr>
<td>TM1 Name</td>
<td>Name of the {TM1/PA} View or Subset.</td>
</tr>
<tr>
<td>TM1 Parent Name</td>
<td>Name of the cube or dimension associated with the {TM1/PA} object.</td>
</tr>
<tr>
<td>*Data DO Field Type</td>
<td>The DO data type of the field. In Figure 6.1, it is a float type. *Only used with {TM1/PA} cube views.</td>
</tr>
<tr>
<td>*Data DO Field Name</td>
<td>The name of the DO tuple field. In Figure 6.1, it is demand. *Only used with {TM1/PA} cube views.</td>
</tr>
</tbody>
</table>
Example:

Refer to Figure 6.1, lines 23 and 44, the \{DO\} variable \textit{demands}. It can be seen that \textit{demands} is an input variable due to the ellipsis. Since \textit{demands} requires data and is used in calculations, then the TM1 Type is a View. For simplicity the cube view name is set as the same as the \{DO\} object name. The name of the cube with this view is called \textbf{Market Demand}.

Since it is a View, the data field information needs to be completed as shown in Figure 6.4. In Figure 6.1, the numerical data in the view correspond to the tuple field \textit{float demand} on line 23. This means the Data DO Field Type is \textit{float}, and the Data DO Field Name is \textit{demand}.

This is also visualized in Figure 6.2, where “Values” are the view data.

\section*{6.5. View Mapping cube}

The cube (and view) name is \textbf{View Mapping} and it contains three dimensions: \textit{DO Objects}, \textit{Dimension Number}, and \textit{View Mappings}. The \textit{Dimension Number} dimension is used to map each dimension of a \{TM1/PA\} view separately, and supports cubes with as many as 10 dimensions. Modelers may add members to support cubes with more than 10 dimensions. The \textit{View Mappings} dimension contains the measures for the view mapping information, which should not be changed.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline

| T1 Dimension Name | DO Field Name | T1 Subset Name | T1 Attribute | DO Field Type | DO Field Name | T1 Element Name |
\hline

<table>
<thead>
<tr>
<th>demands</th>
<th>1 Products</th>
<th>Yes</th>
<th>products</th>
<th>Name</th>
<th>string</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Components</td>
<td>Yes</td>
<td>components</td>
<td>Name</td>
<td>string</td>
<td>component</td>
<td></td>
</tr>
<tr>
<td>3 Version</td>
<td>No</td>
<td>Demand</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Demand</td>
<td>No</td>
<td>Forecast</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>profits</th>
<th>1 Version</th>
<th>No</th>
<th>Forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Time</td>
<td>No</td>
<td>MO1</td>
<td></td>
</tr>
<tr>
<td>3 Products</td>
<td>Yes</td>
<td>products</td>
<td>Name</td>
</tr>
<tr>
<td>4 Profit</td>
<td>No</td>
<td>Profit</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stocks</th>
<th>1 Components</th>
<th>Yes</th>
<th>components</th>
<th>Name</th>
<th>string</th>
<th>component</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Time</td>
<td>No</td>
<td>MO1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Version</td>
<td>No</td>
<td>Forecast</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Stock</td>
<td>No</td>
<td>Stock</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>productionOut</th>
<th>1 Products</th>
<th>Yes</th>
<th>products</th>
<th>Name</th>
<th>string</th>
<th>product</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Time</td>
<td>No</td>
<td>MO1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 Version</td>
<td>No</td>
<td>Forecast</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 Production</td>
<td>No</td>
<td>Production</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\hline
\end{tabular}
\end{table}

Figure 6.4: Cube to define how views are mapped to \{DO\} tuples. Zeroes are suppressed.
Table 6.3: View mapping parameters

<table>
<thead>
<tr>
<th>View Mapping Parameters</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Name</td>
<td>The name of the $n$th dimension in the cube, where $n$ is the current Dimension Number element. Currently determined by {TM1/PA} rule.</td>
</tr>
<tr>
<td>DO Field?</td>
<td>Asking if this dimension maps to a {DO} tuple field (Yes/No).</td>
</tr>
<tr>
<td>TM1 Attribute</td>
<td>Asks which element attribute should be mapped to the tuple field. Can choose “Invariant Name” or another attribute. Follow the attribute suggestions described in Subset Mapping section 6.6.</td>
</tr>
<tr>
<td>DO Field Type</td>
<td>The basic data type for the {DO} tuple field (int, float, string). This is found in the {DO} tuple structure definition.</td>
</tr>
<tr>
<td>DO Field Name</td>
<td>The name of the {DO} tuple field.</td>
</tr>
<tr>
<td>TM1 Element Name</td>
<td>The name of the dimension element to select (slice on).</td>
</tr>
</tbody>
</table>

Only used if DO Field? is Yes

| Only used if DO Field? is No |

6.6. Subset Mapping cube

The cube (and view) name is **Subset Mapping** and it contains three dimensions, similar to the **View Mapping** cube: \{DO Objects\}, \{Attribute Number\}, \{Subset Mappings\}. The Attribute Number dimension is used to map each attribute of a dimension in \{TM1/PA\} separately. Users can add more numbers to this dimension. The Subset Mappings dimension contains the measures for the mapping information.

![Cube to define how subsets are mapped to \{DO\} tuples. Zeroes are suppressed.](image)

Figure 6.5: Cube to define how subsets are mapped to \{DO\} tuples. Zeroes are suppressed.

The big difference between this cube and the **View Mapping** cube is that for views, every dimension must be addressed, whether mapped or unmapped. In this case, the user only needs to list the attributes that are being mapped to \{DO\}, i.e. the attributes present in \{DO\} tuples. In the example above, there is only one field in each tuple, so it’s simply mapped to the “1” Attribute Number dimension member.
Table 6.4: Subset mapping parameters.

<table>
<thead>
<tr>
<th>TM1 Attribute Name</th>
<th>The name of the attribute. See the note on what attributes to use below.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO Field Type</td>
<td>The basic data type for the {DO} tuple field (int, float, string). This is directly in the {DO} tuple.</td>
</tr>
<tr>
<td>DO Field Name</td>
<td>The name of the {DO} tuple field.</td>
</tr>
</tbody>
</table>

**Subset Attributes**

Alias attributes are strongly recommended for text attributes.

If text and numeric attributes map to “key” tuple fields in {DO}, they must contain unique values.

It is also worth noting that if attribute values are ruled into their respective }ElementAttributes cube, then those cells need to be fed as well.

The example above is fairly straightforward with only one attribute each. Below in Figure 6.6 and Figure 6.7 is an example of mapping multiple attributes to a {DO} tuple. Due to the natural structure of dimensions, there is no need to have more than one “key” tuple field when passing a subset. Anything used as a key must be uniquely assigned for each element. Order of the attributes in the Subset Mapping cube does not matter.

```csharp
39  template TProduct {
40      key string product;
41      string id;
42      float weight;
43  }
44  {TProduct} products = ...;
```

Figure 6.6: {DO} tuple structure and variable definition (left), }TM1/PA} subset and corresponding attributes (right).

```plaintext

<table>
<thead>
<tr>
<th>TM1 Attribute Name</th>
<th>DO Field Type</th>
<th>DO Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>products 1</td>
<td>Name</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>2 Product ID</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>3 Product Weight</td>
<td>float</td>
</tr>
</tbody>
</table>

```

Figure 6.7: Subset mapping definition for more than one attribute.
7. Mapping Management

To be able to map a \{TM1/PA\} server to multiple \{DO\} models, all of the \{TM1/PA\} objects listed above in sections 6.3 – 6.6 can be duplicated with different names. However, when building the mapping cubes, the user should keep in mind that all of the dimension structures (i.e. where they are in Rows and Columns) MUST be the same, and they MUST use all of the other same dimensions.

For example, the user may create a dimension called “DO Objects 2,” and a corresponding “DO-TM1 Mapping 2” cube. When creating this \textbf{DO-TM1 Mapping 2} cube, the \textit{DO Objects 2} dimension must be on the Rows and the \textit{Mapping Parameters} dimension must be on the Columns.

There are several dimensions and cubes included to manage multiple DO models as well:

1) \textit{DO Models} (dimension). This is the list of \{DO\} models. The element names are up to the user’s discretion. When running a model through \{DO\}, the name will be passed to the Connector. If only using one \{DO\} model, then only one element is needed.

2) \textbf{DO-TM1 Mapping Management} (cube and view name). This is where users provide the names of the mapping views/cubes and subsets/dimensions for each model.

Object Mapping is the general mapping cube (covered in section 6.4), View Mapping is the mapping cube for views (section 6.5), Subset Mapping is the mapping cube for subsets (section 6.6), and DO Objects is the dimension for the list of \{DO\} variables (section 6.3).

![Figure 7.1: DO-TM1 Mapping Management cube.](image)

<table>
<thead>
<tr>
<th>Table 7.1: Description of the Mapping Management parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>Parent Name</strong></td>
</tr>
</tbody>
</table>
3) **DO-TM1 Control** (cube). Manages the file associated with each {DO} model (Figure 7.2)

![Figure 7.2: Control cube for managing model related files. Relative paths shown.](image)

Table 7.2: Control parameters for each {DO} model. All file paths can be absolute or relative to the {TM1/PA} data files directory.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model File Location</td>
<td>Points to location of the {DO} model file. If <strong>Optimization Method</strong> = DOcloud, then you can optionally enter the path of a folder here, where the folder contains an oplproject. The Connector will add all files from this folder and run the Default Run Configuration, so be sure it contains only files that are appropriate for OPL model types in DOcloud.</td>
</tr>
<tr>
<td>Mapping File Location</td>
<td>The mapping cube information will be converted to a json file and stored in this location.</td>
</tr>
<tr>
<td>Save DO Input File?</td>
<td>User selects whether to save the JSON-formatted data that is sent to DOcloud paired with the {DO} model.</td>
</tr>
<tr>
<td>DO Input File Location</td>
<td>If <strong>Save DO Input File?</strong> = Yes, then this is the path to the input JSON file.</td>
</tr>
<tr>
<td>Optimization Method</td>
<td>One option: 1) DOcloud. The Connector will solve the optimization model on DOcloud and return the results to {TM1/PA}.</td>
</tr>
<tr>
<td>Save DO Output File?</td>
<td>User selects whether to save the JSON-formatted results data that is obtained from {DO}.</td>
</tr>
<tr>
<td>DO Output File Location</td>
<td>This is the path to the output JSON file.</td>
</tr>
<tr>
<td>Connector Run Status</td>
<td>Status of the connector’s last run for this {DO} model. It is either empty, “Error” or “Successful.” See section 13 for what each represents.</td>
</tr>
</tbody>
</table>

Note that the input file should be saved with a small amount of data, possibly for initial testing of the mapping and optimization. When using large amounts of data, saving the file could easily strain memory resources.
8. Building \{TM1/PA\} Subsets and Views

It is the modeler’s responsibility to build the subsets and views.

8.1. \{TM1/PA\} Subsets

Steps:

1) Create a subset of the name specified in the DO-TM1 Mapping cube.

2) Select the desired elements. We strongly recommend that you only include leaf elements. Consolidation calculations can be done in \{DO\}, rather than being passed from \{TM1/PA\}. Also, when data is returned to \{TM1/PA\} from the optimization, consolidated elements cannot currently have data entered.

3) Mapped dimension attributes must be created and populated. Alias attributes are recommended.
   a. The Connector supports the Invariant Name as an attribute, but does not support any other attributes that are automatically generated by Cognos TM1 Performance Modeler (e.g. Index, Weight, etc.).

4) Subsets are primarily passed to DOcloud to enumerate the dimensionality for calculations, the optimization itself, or constraints. Thus, they provide the domain for the problem. Therefore include all elements that are required to optimize the model; if you don’t, DOcloud is likely return an error.

8.2. \{TM1/PA\} Views

Steps:

1) Create a view of the name specified in the DO-TM1 Mapping cube.

2) The only stipulation on where dimensions should be placed is that mapped dimensions (i.e. those that map to a \{DO\} tuple field) must be in the Rows or Columns. For consistency, as well as ease of use and best performance, we recommend placing unmapped dimensions in Context with the required element selected as the slice. Then put mapped dimensions on Rows and Columns to create a consumable view of the cube data.

3) For applying subsets within the view, use the recommendations in section 8.1.

4) The Connector ignores null values in \{TM1/PA\}. This means that the burden is on the \{DO\} model to populate tuple sets with any zero values needed, which can often be done using the subsets that are mapped in section 6.6. See section 9 for an example.

5) Some \{TM1/PA\} modelers may notice that there is almost enough information in the View Mapping cube to create the views automatically through TI processes. You may add members (e.g. Name of Subset) to the View Mappings measures dimension to enable automation of the view creation.

6) You can create objects called Planning Analytics Views in the Planning Analytics Workspace. However, these views are not saved to the TM1 server, and will not be found by the Connector.
Output DO Views

It is important to note that the {TM1/PA} Views that receive any optimized data from DOcloud will have their data overwritten. Thus, we strongly recommend adding a {TM1/PA} measure dimension element specifically for the optimized data returning from DOcloud.

For example, if planning “Production,” then create a measure dimension element called “Optimized Production” to ensure other planning data is not overwritten.

8.3. Versions

We expect that most planning and forecasting models in {TM1/PA} use versioning. Optimization will likely be an iterative process with different scenarios. We recommend retaining different optimization scenarios (input and output data) in different members in a Versions dimension.

9. Building the {DO} Model

The {DO} model must be designed to work with both the {TM1/PA} model and the Connector.

Input and output variables in the {DO} model (i.e. variables with data being transferred between {DO} and {TM1/PA}) can only be sets of tuples. The tuple structure can only contain primitive data types as fields: float, int, or strings. More complex data types are not supported as tuple fields. Output variables must be defined in the post-processing block when using DOcloud.

When naming input and output variables, do not use case alone to distinguish between variables. {DO} is case-sensitive, but {TM1/PA} is not.

The Connector does not pass null values from {TM1/PA} to {DO}. The default null value in {TM1/PA} is zero. If the {DO} model requires null values in certain variables, you will need to ensure that the {DO} model populates those zero values.

For example, assume that we wish to send the time in weeks that is required to transport goods between a production plant and a distribution center (DC). From the New York plant to the Miami DC is one week, so the value of 1 is passed from {TM1/PA} to {DO}. However, from the Miami plant to the Miami DC it is zero weeks. This value of zero would not be passed to {DO}. After the input variable is defined in the {DO} model, the modeler would then add zero values for any of the DC and plant combinations that were not present. One way to populate the unspecified values would be to use an array, which automatically populates any unspecified values with zero. See Figure 9.1.
Another consideration is how to bring separate measures from one {TM1/PA} view into the same tuple. The Connector is only able to map a {TM1/PA} dimension (not individual elements) to a {DO} tuple field.

For example, assume that there is a cube with two dimensions, a Plants dimension and a Measures dimension. This Measures dimension contains measures such as “Working Days” (weekly basis) and “Hours per Day” that the plant is operational. In the DO model, we would want three tuple fields: one for the plant name, one for working days, and one for hours per day. However, the Connector does not allow us to simply map these separate elements from the Measures dimension to multiple tuple fields.

SOLUTION: See Figure 9.2. In this case, you would map the {TM1/PA} Measures dimension from the view (with a subset containing the two desired measures) to the measure tuple field in the Measure1 tuple structure. Then in the DO model, reformat it to the desired tuple structure (Measure2). The variable m1 in Figure 9.2 would be the input data from {TM1/PA}, and m2 would be the same data reformatted within the {DO} model to a more convenient tuple structure.
10. Running the Connector using TI Processes

The entry point for running the Connector with multiple users is the queue management system, described in section 11, specifically the *DO Run History Update* process.

The *Run Optimization* TI process uses Java extensions to run the Connector jar file. The only parameter this process needs is the name of the corresponding element name in the *DO Models* dimension of the *DO* model being run. You should be aware that the process will often end by saying it completed successfully, whether it did or not. This is because it is running the Connector outside of *TM1/PA*. For an actual status, look at the Connector Run Status in the *DO-TM1 Control* cube (see Figure 7.2).

The *Clean Up Optimization* TI process is optional to delete the views and subsets specified in the mapping cubes. This is only recommended to use if you have also created an associated TI process to create the views and subsets prior to running the Connector. We generally recommend keeping the subsets and views as static as possible. This process also requires the element name in the *TM1/PA* *DO Models* dimension associated with the *DO* model.

11. Queue Management

The Connector is single-threaded, which requires a queue system to allow multiple users to submit requests to run the Connector while it may be running. The queue system includes a chore, several processes, cubes, and dimensions.
If only one user is running the Connector, then the queue mechanism is not necessary. You can directly use the Run Optimization TI process, as discussed above in section 10.

11.1. Add a DOcloud run to the queue

It is recommended to create a cube where the user can select which model they want to run from a picklist of DO Model dimension elements. Then pass the model name and user name to the DO Run History Update process, which will add the model and user to the DO Run History cube, and indicate it is ready to run.

![Parameters](image)

Figure 11.1: DO Run History Update process parameter entry example.

11.2. Chore runs each queued optimization

The chore DO Ready Run Chore will run every 5 minutes once activated. It locates each of the queued runs in the DO Run History cube, and runs each (one at a time).

The DO Queue Control cube is maintained by processes to detect when an optimization is already running.

![Figure 11.2: View for DO Queue Control cube.](image)

The DO Run Control cube maintains who has run each model and shows what the next run will be for each user and model combination. See examples of these cube views in Figure 11.2.
12. Final \{TM1/PA\} Subsets and Views for the Example

Figure 12.1: Subset for *products* \{DO\} input variable (dimensionality).

Figure 12.2: Subset for *components* \{DO\} input variable (dimensionality).

Figure 12.3: View for *demands* \{DO\} input variable.

Figure 12.4: View for *profits* \{DO\} input variable. Based on the View Mapping, only the “Profit” values will be mapped to \{DO\}. Note that the Connector supports working subsets as well as named subsets.
Figure 12.5: View for stocks {DO} input variable.

Figure 12.6: View for productionOut {DO} output variable. The optimization has not been run yet, so no production values are available.

Figure 12.7: View for productionOut {DO} output variable after the optimization was run. The next run of the DO Connector will replace these values.

13. Troubleshooting

When the Connector is run, a log file named connector.log is created in the same directory as the Connector jar file (<TM1 Server data>/javaextensions/user/). This log file contains informative error messages. As an example, if a View was not created, the error message at the end of the log file will tell the user that it cannot find a View. Any errors will generally show at the end of the log file.

After the Connector has been run, the DO-TM1 Control cube will show the “Connector Run Status.” This can be any of the following values:

1. Empty (nothing in the cell): an error has occurred, and you should check the connector.log file. This indicates a problem with simple initialization of the Connector.

2. “Error”: an error occurred while the Connector was running. Check the connector.log file, as this could be many different things.

3. “Successful”: the Connector ran successfully and had no errors.
The `optimization.log` file provides the details of the optimization from DOcloud, including potential errors. This log file is in the same folder location as the Connector jar file.

Support for the Connector is NOT guaranteed. IBM expects to monitor the [IBM developerWorks Community Forums](https://www.ibm.com/developerworks/community) for questions and discussion of the Connector. Please report any potential bugs and defects in the forums.
The Samples are owned by IBM and are copyrighted and licensed, not sold. They are licensed under the terms of the International License Agreement for Non-Warranted Programs, which you accepted prior to downloading the content from IBM developerWorks web site (https://www.ibm.com/developerworks/).

See the complete license terms at:

By using the Samples, you agree to these terms.

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