User Guide | PUBLIC
Document Version: 2204a – 2022-04-19

Thing Modeler Apps (OData version)

OData Version

THE BEST RUN SAP

Content

1 Modeling Things: Overview. 3
1.1 Feature Availability. 7
1.2 Migrating from REST to OData. 7

2 Packages: Overview. 10
2.1 Create a Package. 12
2.2 Maintain a Package. 14
2.3 Delete a Package. 15

3 Thing Modeler: Overview. 16
3.1 Create a Thing Type. 18
Thing Onboarding. 21
Requirements for Successful Thing Onboarding. 24
3.2 Maintain a Thing Type. 25
3.3 Delete a Thing Type. 26
3.4 Create a Thing. 27
3.5 Maintain a Thing. 30
3.6 Delete a Thing. 31

4 Thing Properties Catalog: Overview. 33
4.1 Create a Property Set. 34
Additional Settings for Numeric Values. 38
4.2 Maintain a Property Set. 40
4.3 Delete a Property Set. 41

1 Modeling Things: Overview

Overview of the different steps of thing modeling

Introduction

Thing Modeler is the name of a group of apps that serve the purpose of covering all aspects of tasks to be done during onboarding of things to the SAP IoT platform. Here, a thing is seen as the digital twin of a real-world asset (such as a building, machine, or vehicle). To foster efficiency gains, the underlying architecture encourages you to set up basic entities with a high reuse potential. To accomplish this, you typically do not start with an object called "thing". Rather, the system expects that you think about reusable sets of properties that are typical of a particular type of thing before you start instantiating concrete things. The starting point for all these activities, however, is the definition of so-called packages. A package serves as a container for thing types as well as their properties.

The technical aspects that are covered by the *Thing Modeler* apps comprise the following:

- Defining packages serving as containers for thing types and things.
- Defining reusable sets of properties (so-called property sets)
- Defining thing types to which a property set can be assigned
- Deriving things from thing types
- Onboarding things, that is, connecting them to the sensors of real-world objects.

> **i Note**
>
> In addition to the technical aspects listed above, there is even more to thing types and things that goes beyond the functional scope of the *Thing Modeler* apps. For example, events and thing hierarchies cannot be defined here. If you want to use such advanced features, you can accomplish this programmatically via the respective APIs as described in the API reference documentation. For more information, see What is SAP IoT.

Apart from these technical aspects, there is a second area of system administration: The definition of an organizational framework that comprises companies, persons and users, user groups, and authorizations. For this type of administrative tasks, SAP offers a second group of apps, namely the *Tenant Administration* apps.

Cross-App Relationships

Although the *Thing Modeler* apps are constructed as self-contained, stand-alone applications, several connections exist between them. These connections illustrate that the different apps can be seen as different perspectives on the same subject matter, namely the technical aspects of thing modeling. For example, from the *Thing Modeler* app, you can navigate to the property sets assigned to a thing type in the *Thing Properties Catalog* app. Also, when you derive a thing from a thing type, the system demands that you enter a valid authorization group that you have prepared in the *Object Authorization* app, which belongs to the *Tenant*

Administration app group. Also, with packages as the basic container element for both thing types and properties, the *Packages* app lets you navigate directly from a particular package to both *Thing Modeler* as well as *Thing Properties Catalog*. The current package is then automatically passed on to the target apps as their default repository for the session.

Roles and Target Groups

The separation between apps for organizational versus technical aspects of tenant administration is based on the assumption that in most cases, persons with different skill sets will take care of these different aspects. While the apps for organizational aspects have managers or administration experts as their target audience, the Thing Modeler apps are addressing a more technical audience of engineers. Consequently, SAP offers different role templates with predefined authorizations that are needed to work with one of the two app groups. You can use these role templates as a blueprint for setting up custom roles according to the needs of your company.

Limitations

Although the system does not enforce any hard-coded limitations with respect to the number of objects that you can set up with the Thing Modeler apps, we recommend keeping the complexity within a package at a moderate level. This helps you to avoid any potential problems with the database or any other component of the IoT Application Enablement system landscape. As a rule of thumb, we recommend **not** to exceed the following limits:

- 10 thing types per package
- 10 property sets per package
- 10 properties per property set

To compensate for the recommended limits listed above, there is **no** limit to the number of packages that you can set up in each tenant. So if you need a high number of thing types or property sets for your use cases, simply distribute them over a set of multiple packages with *Tenant* scope. Likewise, if you need more than 10 properties per property set, just split up your property sets and distribute the properties over a bigger number of property sets.

> **i** Note
>
> If, for whatever reason, you feel you cannot live with the recommended numbers listed above, you may decide to go beyond these limits (but at the risk of performance issues). However, there is an ultimate limit to the number of properties in a property set that you cannot exceed: The maximum number of properties used to hold measured values in a property set is 500.

Language Handling in the Thing Modeler Apps

General

The Thing Modeler as well as the Thing Properties Catalog apps support texts in different languages for the objects that you create with the help of the SAP IoT apps, for example, thing types or things. With that, you are free to maintain object descriptions in your preferred language.

> **i** Note
>
> The language handling described here is in effect only for the *Description* fields of the various entities supported by the apps. As opposed to that, there is no language support for the technical names of all these objects. In addition, there are even strict naming conventions for technical names that must be observed to ensure the consistency of the stored data.

Language Handling in Detail

When you create or maintain an object with one of the Thing Modeler apps, the app takes care of the browser language settings that are in effect. Here, the following scenarios of language determination can occur:

- Explicit language setting: You can force the apps to run in a particular language mode by adding the `sap-language` parameter to the URL of the app. For example, with `sap-language=DE`, you force the app to run in a German language environment.
- Implicit language setting: If there is no language requested explicitly via a URL parameter, the app evaluates the language preference settings of the browser in which the app is running. If the preferred browser language is one of the supported languages of SAP IoT, the app switches automatically to that language.

Once the app has determined the language and adapted itself to that language, all object descriptions that you maintain for newly created objects (or for existing objects where no description has been maintained so far) are stored in the system together with the ISO code of the determined language. Of all the supported languages, English is treated as a fallback language that the system uses in all cases where otherwise the *Description* field would remain empty. This results in the following scenarios:

Language Resolution for Object Texts

	Browser Language		
Object Created in	**EN**	**DE**	**FR**
EN	EN	EN or DE	EN or FR
DE	(-)	DE	(-)
FR	(-)	(-)	FR

As you can see in the table, for an object that has been created in a session with browser language EN, the description text is stored with the language code `EN` assigned. With English being the system fallback language, the English text is also displayed in a browser session in a different language like German or French. However, this behavior can be overwritten by entering a description that is specific for the current browser language.

In contrast to that, if an object has been created in a session with browser language other than English (for example, German), the description text is stored with that language code assigned. In this case, the description in all other languages (including English) is empty.

Limitations

Although the ability of the system to handle object description texts in multiple languages in parallel is a powerful feature and especially useful for cross-country organizations, it can also lead to situations where the system seems to behave unexpectedly at first sight. This situation may arise when an object has been created in a session with a browser language other than English. As you can see in the table above, this leads to the consequence that the *Description* field is empty for all languages other than the creation language.

Also, the UI texts of the apps are only available in English and German, with English being the fallback language in case the browser settings call for a language other than these two. In that case, you might be tempted to maintain descriptions in English, too. In that case, however, these English descriptions would be stored with the language flag that corresponds to the browser settings (for example, FR or ES). As a consequence, such descriptions would **not** be displayed when the app runs in a browser with the language set to EN.

> **i** Note
>
> All language handling mechanisms described above are based on the assumption that the creation language is among the 40 languages that are supported by the SAP IoT framework. If the language setting of your browser refers to any other language that is **not** part of the set of supported languages, the system raises an error message.
>
> For the list of supported languages, see ISO Language Codes.

SAP Fiori Launchpad

The apps that are provided with SAP IoT are designed for, and based on, SAP Fiori. The various apps are grouped into two main areas, *Tenant Administration* and *Thing Modeling*. Each single app has its own tile on the Fiori launchpad. For more information on how to get the best out of the SAP Fiori launchpad, see the SAP Fiori Launchpad user guide.

Related Information

Packages: Overview [page 10]
Thing Modeler: Overview [page 16]
Thing Properties Catalog: Overview [page 33]

1.1 Feature Availability

Product and feature availability per deployment landscape

Overview

Over the course of time, the Thing Modeler app suite has been adapted to changes in the underlying basis technologies, such as UI model or the SAP IoT services for SAP BTP service. Not all of the subsequent technologies can be offered simultaneously within one and the same cloud software instance. However, to serve our customers' needs, we have decided to make different feature sets available which, for technical reasons, must be separated against each other. This separation is accomplished by deploying our solution with different feature sets to different cloud landscapes. This results in the product and feature matrix as given in the following table (note that the `cf-eu1` landscape is **not** available for new customers):

Feature Availability

	Landscape		
Feature	**Open Stack (cf-eu1)**	**AWS (eu10)**	**AWS (us10)**
OData-based apps	X	X	X
Flexible sensor mapping on thing type level	X	X	X

> **i** Note
>
> SAP IoT offers OData-based apps for modeling things, creating users, and so on. When exploring the platform, you may discover additional apps that are not explained in product documentation.
>
> For these undocumented apps, SAP does not offer any support whatsoever, nor does SAP give any warranty with regard to the future availability of such apps. We therefore highly recommend not using any undocumented apps.

1.2 Migrating from REST to OData

Explains the necessary steps to migrate from the REST-based to the OData-based version of Thing Modeler.

Overview

> **i** Note
>
> This chapter is relevant only for users who have already been using Thing Modeler before version 1.65 (September 2018).

Introduction

Almost since the very beginning of SAP IoT, there have been API services provided that made use of two different HTML flavors, either REST or OData. Up to version 1.64, Thing Modeler could only handle thing types, things, and property sets that had been created in packages that were configured via the REST API services.

Starting with version 1.65, SAP IoT provides two versions of Thing Modeler:

- One for REST-based thing models (identical to the previous version).
- One for OData-based thing models.

We highly recommend using the new OData version of Thing Modeler, as this is the only version that will see future enhancements and new features. However, before you can switch from the REST version to OData, two things need to be taken care of:

- Your existing thing models need to be migrated from REST to OData. This is done for you by SAP upon your request.
- You need to exchange the Thing Modeler tile in the Fiori Launchpad so that the tile correctly points to the OData-based version of Thing Modeler. This step must be taken by yourself or by your administrator, respectively.

> **i Note**
>
> The migration is necessary because thing models created with the REST-based API services are incompatible with OData thing models and vice versa. Also, each thing model flavor can only be used with the Thing Modeler version dedicated to the respective flavor.

Replacing the Thing Modeler tile in Fiori Launchpad

The following steps are a subset of the complete tenant onboarding that is necessary for each SAP IoT customer. While each new customer is automatically guided through this procedure by the Quick Start guide, existing customers that have already been onboarded to an earlier release now need to take these steps again to modify the Fiori Launchpad tiles for the Thing Modeler apps.

Defining Roles for Thing Access and Tenant Administration

The following steps are necessary because using the OData version of Thing Modeler requires assigning a new role to users who are supposed to work with Thing Modeler (OData version).

1. In the SAP BTP Cockpit navigation pane for your SAP IoT subscription, choose *Security* > *Roles*.
2. Choose *New Role*.
3. Enter a Role Name. We recommend aligning the role name with the name of the role template to be assigned in the next step: **`sapdev_iotas_Thing_Engineer_Odata`**
4. From the list of role *Templates*, choose the one for the protocol you want to use: `iotas_Thing_Engineer_Odata`
5. In the *Attributes* section, define an attribute with the following settings: *Attribute Name* = **`siteGroupRoles`**, *Source* = **`Static`**, *Value/SAML Attribute* = **`Thing_Engineer_OData`**
6. Choose *Save*.

Creating a role collection

In the following section, you set up a new role collection for the previously created role. Although it would also be possible to use the existing role collection for the Thing Modeler apps, we recommend using a new dedicated role collection for the OData-based version of the apps. This allows for exactly tailoring the set of

apps to the needs of the users and avoids confusion that might be caused by seeing identical tiles for both REST as well as OData.

1. In the SAP BTP Cockpit navigation pane, choose *Security* > *Role Collections*.
2. Choose *New Role Collection*.
3. Define a *Name* for the new role collection, for example **`iotae_role_collection`**.
4. Choose *Save*.
5. Select the new role collection by clicking its name.
6. Choose *Add Role* to add the thing engineer role to the role collection.
7. Choose an *Application Identifier* and choose the `iotas_Thing_Engineer_Odata` *Role Template*.
8. Add the thing engineer role to the role collection.
9. From the list of roles, choose the role you have previously created based on the `iotas_Thing_Engineer_Odata` role template.
10. Choose *Save*.

Assigning Role Collection to User

The last few steps are required to make the new OData-based apps available to the user.

1. In the SAP BTP Cockpit navigation pane, choose *Security* > *Trust Configuration*.
2. Choose *SAP ID Service*.
3. Enter your *User* (typically, this is your e-mail address).
4. Choose *Show Assignments*.
   The system notifies you that there are currently no assignments.
5. Choose *Add Assignment*.
6. Assign the previously created role collection to your user.
7. Choose *Add Assignment*.

Related Information

Providing Authorizations

2 Packages: Overview

Overview of the Packages app

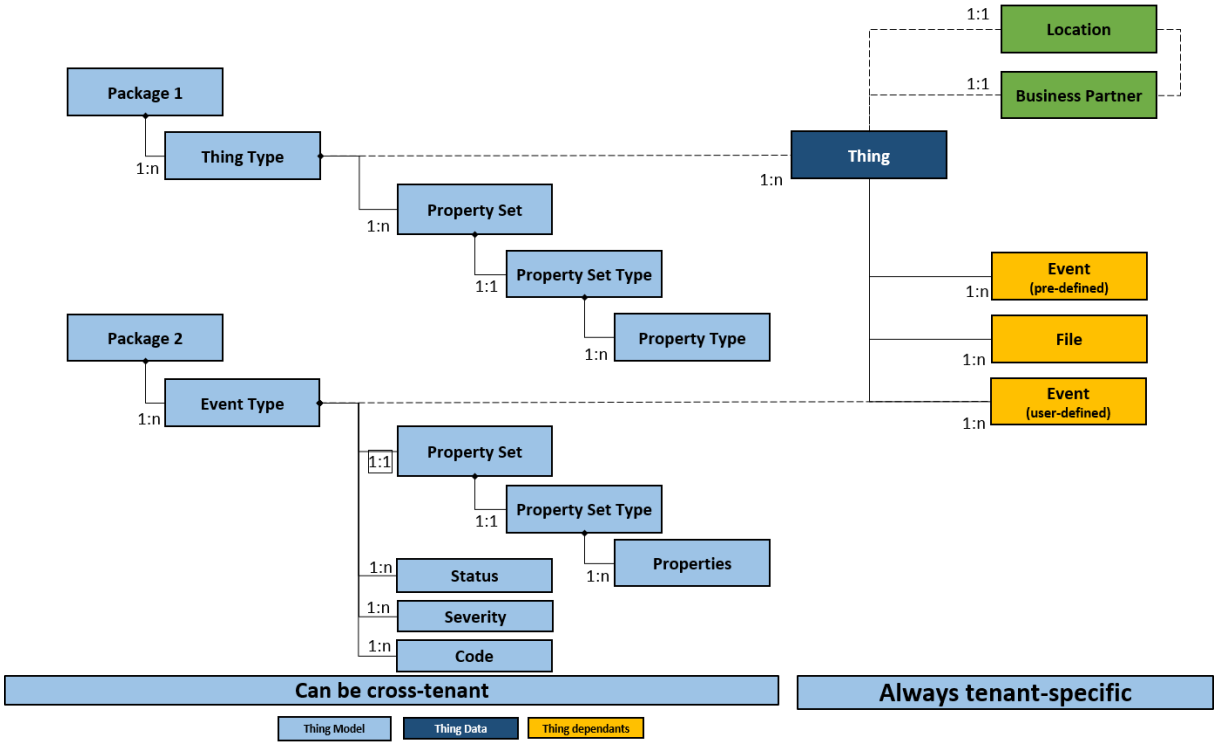
Overview

When it comes to modeling thing types and things, the first thing to start with is to define a package to which new thing types and things shall be assigned. For this, the *Packages* app is the tool of choice.

A package serves as a container for thing-related objects, such as thing types, things, property sets, properties, or event types. For objects playing the role of metadata entities (for example thing types or event types) it is possible to make them available in the entire system, that is, beyond the borders of the tenant to which a package belongs. This is useful if customers who are assigned to different tenants are using tools and equipment of the same kind. An even more important scenario would be a platform provider who decides to make certain basic thing models available across the entire platform and all tenants so that platform customers can use these models as a blueprint for their own thing types and things.

> **Example**
>
> Let's assume one software platform where two different car manufacturers have been onboarded independent of each other. Accidentally, both manufacturers happen to use the same type of welding robots in their production lines, and they want to monitor these machines in their individual environment inside of the IoT application platform. In this situation, if the particular type of robot has already been set up as a digital twin before (that is, as a fully specified, but still abstract, thing type), the cloud platform provider could offer newly onboarded customers to make use of this already existing thing type for their own purposes. However, as a prerequisite for such a service offer, that thing type must have been defined in a package that can be accessed across the borders of an individual tenant.

The following image gives a good overview of all the objects that may belong to a package and how they are related (note that in this technical architecture diagram a terminology is used that slightly differs from what you see in the user interface of the *Packages* app):



Note that in the above image, you can easily differentiate between the abstract object types (left-hand side) that can potentially be used across tenant boundaries, as opposed to the concrete, physical objects on the right-hand side that are always isolated within the tenant in which they are created. This reflects the simple and obvious fact that a robot of a particular type, which is installed in a production plant of company A is not identical with another robot (though of exactly the same type) that is installed in a production plant of company B.

Package Versioning

When you work with packages, you soon notice that with every single change that you apply to a package, the package version number is automatically incremented. This mechanism has been implemented in the backend to ensure that even the slightest modification of a package can be traced back to a package version that is totally identical to its predecessor version with the only exception of that single modification that triggered the version increment. The purpose of this mechanism is to make sure that a package always remains in a consistent state. However, should an error occur, it is always possible to fall back to a previous state of the package with the highest possible similarity to the package version that caused the error.

The version numbering scheme uses a decimal notation that consists of three hierarchical parts, separated by the dot ".". The initial version of a newly created package is "1.0.0". With each change, the lowest significant part is incremented by 1, with a range from 0 through 999. Once a version number part has reached the maximum value of 999, the next change leads to a reset of that part back to 0, while the version number part with the next higher significance is increased by 1.

> **Example**
>
> - 1.0.998
> - 1.0.999
> - 1.1.0
> - 1.1.1

> **Note**
>
> Versioning of packages is an entirely automated process. There is no way for a user of the Thing Modeler apps to influence the versioning mechanism or to manually set a particular version for a package.

Navigation to Other Apps

As already mentioned, setting up packages is a prerequisite for all other activities that are related to modeling things. It is therefore natural that once you have defined the packages you need, you want to proceed with setting up thing types, things, and their properties. To accomplish this, the *Packages* app supports direct navigation to both the *Thing Modeler* as well as the *Thing Properties Catalog* app. In the list of packages that is shown on the entry screen of the app, you find the following options for each package:

- Choose to open the *Thing Modeler* with the selected package.
- Choose to open the *Thing Properties Catalog* with the selected package.

2.1 Create a Package

Process steps for creating a package

Context

As an administrator, you want to define a package which will then serve as a starting point for creating thing types and things as well as their properties.

When defining a package, it is important to choose the appropriate scope level. With the scope, you decide about the visibility of all objects that are contained in the package. The following scope levels are available (sorted by increasing visibility):

- Private: Objects contained in a private package can only be accessed by objects belonging to the same package. This is the default setting.
- Tenant: Objects contained in a tenant package can be accessed by objects belonging to all packages of the same tenant.
- Public: Objects contained in a public package can be accessed by all objects belonging to any package, across all tenants. This setting provides the highest reuse potential, but also the risk of unwanted side effects in case of later modifications.

> **Note**
>
> Public packages may exist in the system, but you cannot create them in the Packages app. You have only read access to objects in a public package that belongs to a different tenant than the referencing package.

> **Note**
>
> You can create packages only in the tenant to which you are currently logged on. Logging on to a tenant is an implicit step that is performed by the system when you log on to the Fiori Launchpad, from where you start the Packages app. The tenant that was used for that is then automatically used as a prefix for the package name and cannot be changed.

Procedure

1. From the launchpad, start the *Packages* app.

   The system presents the list of available packages.
2. In the upper right corner, choose + *Add*.

   The system adds a new row to the list of packages.
3. Enter a unique technical *Name* for the new package (required) and a short *Description* of the package's purpose.

   For the technical name of a package, the following requirements apply:

   - A package name must always start with the name of the tenant to which the package belongs, followed by a dot ".". For more information on tenant names, see the Tenant chapter in the API reference.

     > **Note**
     >
     > The tenant name is automatically determined and used as a package name prefix by the system internally. Inside of the Packages app, the tenant prefix is **not** displayed. If the tenant name contains a hyphen ("-"), the system automatically replaces the hyphen by a dot (".") in the package name.

   - The name may only consist of characters from the following character set: [a..z0..9] plus the dot ".". Upper case letters as well as all kinds of special characters are **not** allowed.
   - An alphanumeric sequence between two dots must consist of two characters or more. The same applies for the sequences at beginning and at the end of the package name.

   > **Example**
   >
   > Here are some examples of valid package names:
   >
   > - `test.sandbox.package.cars`
   >   This is package "package.cars", which belongs to a tenant called "test-sandbox". Note that the hyphen in the tenant name must be replaced by a dot.
   > - `sap.iot.vehicles40`
   >   This is package "vehicles40" in tenant "sap-iot".
   >
   > Again, keep in mind that the tenant used as a prefix of the package name is visible and relevant only on database level and for programmatic access to a package.

4. Decide about the visibility of the new package and the objects it contains by choosing the *Scope* that fits your needs best.

   During creation, you can only choose between the scope levels **`private`** and **`tenant`**. You cannot create a public package with the Packages app.

5. Once you are finished with defining the package, choose *Save* to save the package.

Results

The new package is now available in the system and can be used as a starting point for creating thing types and things as well as their properties. For technical reasons, new packages are always created with a default property set assigned. You are free to use this default property set in the *Thing Properties Catalog* and adjust it to your needs, or to create your own property sets and delete the automatically created property set later.

2.2 Maintain a Package

Process steps for maintaining packages

Context

As an administrator, you want to modify an existing package.

> **i** Note
>
> You cannot change the technical *Name* of a package.

Procedure

1. From the launchpad, start the *Packages* app.

   The system presents the list of available packages.

2. From the list of packages, choose the one you want to maintain. If you don't see the package that you want to change, use the *Search* to narrow down the number of packages presented in the list. The search function always searches the complete set of packages available in the system, regardless of whether a package has been loaded into the UI or not.

3. Once you have found the package you want to change, choose *Edit Package* .

   The package is now in edit mode and you can change its values as desired.

4. Modify the field values according to your requirements.

5. Choose *Save Package*

   You can only save the package if its attribute values have changed compared to what is already stored in the database.

2.3 Delete a Package

Process steps for deleting packages

Context

As an administrator, you want to clean up the system by deleting packages that are no longer used and that do not have any thing type or event type assigned.

> **i** Note
>
> You should keep in mind the following aspects:
>
> - A package that is used by other packages cannot be deleted.
> - A package can only be deleted if there is no relationship between the package and any other entity in the database. Such entities include thing types, event types, property sets, as well as other packages that may be used by the current package.
> - If you want to delete a package that has still certain objects assigned, the only way to delete the package is to delete all these assigned objects first. However, since a package serves as a container for a typically high number of objects that are crucial for an IoT model, deleting a package should be considered only in rare, exceptional cases. We highly recommend aligning yourself with all potentially involved stakeholders before you delete a package.

Procedure

1. From the launchpad, start the *Packages* app.

   The system presents the list of available packages.
2. From the list of packages, choose the one you want to delete. If you don't see the package that you want to delete, use the *Search* to narrow down the number of packages presented in the list. The search function always searches the complete set of packages available in the system, regardless of whether a package has been loaded into the UI or not.
3. Once you have found the package you want to delete, choose *Delete Package*.

   The system presents a confirmation dialog where you have to confirm your decision to delete the package.
4. Confirm your decision.

3 Thing Modeler: Overview

Overview of the Thing Modeler App

Introduction

When it comes to managing your physical assets (buildings, vehicles, machines of all kinds), it is obvious that one of the major preparatory tasks is defining the things you are interested in, that is, the digital twins of each of your assets. Once you have set up all these things within SAP IoT, you have laid the foundation of your own share of the Internet of Things.

Basic Principles

You use the Thing Modeler to create and maintain two different types of objects, things and thing types:

- Things
  A thing in the Thing Modeler is what is often referred to as the "digital twin" of any kind of a real-world object, or a virtual representation of a physical asset. A thing is almost completely defined by the settings that are in effect for the thing type from which it has been derived. Only a few (but nevertheless important) settings can be made directly for one individual thing.
- Thing Types
  A thing type describes the inner structure of a group of things of the same kind. The major aspect of a thing type is the set of properties that all things derived from that type have in common. These properties are collected in reusable sets of properties called *Property Sets*. These property sets can contain basic data, measured as well as calculated values, and status information. For more information, see Thing Properties Catalog: Overview [page 33].

The relationship between thing types and things is similar to the one you may know from object-oriented programming languages as the principle of inheritance: A thing that you derive from a thing type inherits all the properties that have been defined for the type, so that the inner structure of all things derived from the same type is identical. In addition, certain aspects may be different for each individual thing. This reflects the real-world phenomenon of a particular product type (for example a car like the famous VW Beetle) where millions of individual cars follow basically the same construction plan but differ in details like color, motor type, tire manufacturer, split or non-split rear window, and so on.

Thing Types and Things: What Belongs Where?

Here is a short overview of the features of both thing types and things:

- Thing Type
  - Unique Name (cannot be changed after creation)
  - Description (may be used for manufacturer-related information such as construction series, year of construction, and so on)
  - Assigning property sets with definitions of basic data, measured and calculated values, and status values
  - Default image for all derived things

- Thing
  - Unique Name (cannot be changed after creation)
  - Description (may be used for customer-specific inventory number, location information, and so on)
  - Individual image
  - Authorization group
  - Location data
  - Provider data needed for acquiring time series data related to a particular thing

Thing Types and Things: Where to Start?

Although thing types and things are in a mutual dependency relationship (a thing type without things is useless in daily business, and a thing without a thing type assigned must be counted as undefined and makes no sense), the Thing Modeler still gives you some freedom for your decision where to start the modeling process. You have the following options:

- Start with thing types: Once you have completed the definition of a thing type, you can proceed by clicking *New Thing* in the details area of the thing type UI. The system asks you for some basic data and then creates a new thing based on the currently selected thing type.
- Start with things: If it feels more natural for you to start with creating the things (because you have a strong sense for their character as the digital twins of their corresponding real-world objects that you can see and touch when you walk around in your facilities), then you can start right away with that. This approach, however, requires that the respective thing types are already present in the system because assigning a thing type to a new thing is mandatory.

Mass Creation of Thing Types and Things

In daily business, it is common that both thing types and things are required in higher numbers with relatively small differences between each instance of a type or thing. This is especially true for things, as it is totally normal to have many identical instances of a particular asset (for example, a certain type of a wind turbine or a gantry crane of which you have dozens of copies in productive use). Here, the *Copy* function comes in handy, which is available for both thing types and things. With this function, you can create an exact copy of an already existing thing type or thing. Once you have created the desired number of identical objects, you can walk through the row of objects and make some individual adjustments, if need be.

Personalizing the Thing Modeler

For thing types and things, the Thing Modeler offers various screens where certain details of the thing types or things are presented in tables. For example, there are tables for basic data, measured values, and calculated values of a thing type or thing. You can change the look and feel of these tables by clicking the ⚙ *Settings* button that you find on top of each single table. The system then presents a dialog box where you can make the following settings:

- Decide which columns shall be displayed and which not.
- Change the sequence in which the columns are displayed.
- Define filter conditions for the data to be displayed in the table.
- Change the sort order for the table content.

Related Information

Thing Properties Catalog: Overview [page 33]

3.1 Create a Thing Type

Process steps for creating a thing type.

Context

As an administrator, you want to define a thing type that can later be used as a blueprint for individual things to be derived from the type.

> **i** Note
>
> When creating a thing type, the following aspects are worth knowing:
>
> - You can create thing types only in packages where you have write permission. Packages where this is **not** the case (that is, you are only authorized to display existing thing types) are flagged with a 🔒 symbol. Also, packages with `public` scope that have been created by another user are always read-only for all other users.
> - The property sets that you assign to a thing type can only be changed as long as no things have been derived from the type.

Procedure

1. From the launchpad, start the *Thing Modeler* app.

   The system presents the list of thing types available in the preselected package and loads the details of the first list entry into the details area. If you want to create a thing for a thing type in a different package, choose the package from the *Package* dropdown list.

2. At the bottom of the list of thing types, choose + *Add*.

   The system displays the *Create New Thing Type* dialog box.

3. Enter a unique technical *Name* for the new thing type (required) and a short description of the thing type's purpose.

   > **i** Note
   >
   > A thing type name must always start with the package name to which it belongs, followed by a colon ":". For more information, see Create a Thing Type in API Reference Guide.
   >
   > The package name is prefixed by the system internally. Inside of the Thing Modeler app, the package prefix is not displayed.
   >
   > However, in the URL, you can find the fully qualified name of the thing type.

4. Once you are done, choose *Save*.

   The system presents the details screen for the new thing type.

   - *Basic Data*: The system displays an automatically created property set `DefaultImagePropertySet` with one default property `DefaultImageProperty`. This property set is needed for internal technical

reasons in case you want to provide a predefined image to be displayed for all things that you later derive from the thing type. If you don't need such an image, you can as well delete the default property set. You can assign your own property sets and modify and add properties whose values are supposed to be stable over time (like length, width, weight, voltage, and so on).

- *Measured Values*: You can assign properties that represent values that have been determined by sensors that are attached to a thing.
- *Calculated Values*: You can assign properties that represent values that have been determined by sensors and have been aggregated and calculated by a preprocessing engine in the backend.

> **i Note**
>
> For both measured values as well as calculated values you can also define status values that may - but do not have to - correspond to the measured or calculated values that you define in the scope of a particular property set.

5. You can now define the inner structure of the new thing type by assigning one or more property sets to the different sections (*Basic Data*, *Measured Values*, *Calculated Values*).

   In each section, when you choose + *Add*, the system automatically restricts the list of available property sets to those matching the property set type for the corresponding section. If a suitable property set does not yet exist, you can easily navigate to the *Thing Properties Catalog* and create a new property set according to your needs. After that, you may return to the *Thing Modeler* and assign the property set to the new thing type.

   The property sets that the system offers for selection are always part of the currently selected package, or of its subordinate packages (if there are any).

6. You may decide to add an image to the thing type data.

   This image is used not only for the thing type itself. It is also used as a default image for the things that you later derive from the type. For images, both PNG and JPEG format are supported.

   > **i Note**
   >
   > Please be aware that an image that you assign to a thing type or a thing may contain any kind of visual information. There is **no** mechanism in place to check the image content in an automated fashion. It is therefore up to you to ensure that all images uploaded to the system do comply with legal requirements and do not unveil any confidential information.

7. You may now proceed to the *Connectivity* tab and define a mapping between the sensor types of the available physical devices and the corresponding properties of the thing type. These mappings are later inherited by each thing that you derive from the thing type. To accomplish this, proceed as follows:

   a. In the *Connectivity* section, choose + *Add New Mapping* to define a new type mapping.

      The system displays the *New Mapping* screen.

   b. Enter a *Name* and a *Description* for the new mapping.

   c. From the list of available *Sensor Types*, choose one or more sensor types.

      In the *Thing Model or Device Model Mapping* table, the system displays the property sets of the thing type with their properties.

   d. For each thing type property, choose from the matching *Device Properties* the one you want to use in the mapping.

   e. Choose *Save* to save the mapping.

You may repeat the previous steps to define additional mappings for the thing type. When it later comes to instantiating a thing from the thing type, you can choose from the available mappings which one shall be used by a particular thing.

8. Once you are finished with defining the thing type, choose *Save* at the bottom of the screen to save the new type.

Results

Now that you are done with defining a thing type or thing, let's have a look at the list of properties that you have assigned to it: The properties are presented in a hierarchical list that consists of either two or three levels, depending on the optional settings that have been made for a property in the *Thing Properties Catalog*:

Level	Object	Description
1	Property Set	An object that serves as a container for a set of individual properties. There can be one or more property sets assigned to a thing type or thing.
2	Property	The properties representing master data (*Basic Data*), sensor data (*Measured Values*) or data derived from sensor data (*Calculated Values*).
3	Reference Property (optional)	This level is only relevant for properties where addditional optional settings have been defined in the *Thing Properties Catalog*, such as thresholds, target value, or a unit of measure. If these additional settings have been defined, the system reflects that by automatically generating a reference property set in the backend. The reference properties are accessible via a respective API service method, which enables you to support scenarios where timeseries data for the same property is collected in different units of measure for different things. You can then use the reference property to recalculate the values so that they all refer to the same unit of measure and can be compared and analyzed in a useful manner (see Create a Property Set [page 34] for an example). If reference properties have been generated, the Thing Modeler displays them in the list of properties on level three. For each reference property, you can find its technical name that you need to know in order to access it via the API. For more information, see Reference Property Set Type.

Related Information

Create a Thing [page 27]
Create a Property Set [page 34]
Thing Onboarding [page 21]
Requirements for Successful Thing Onboarding [page 24]

3.1.1 Thing Onboarding

Establishing a connection between a physical device and a virtual thing

Overview

When setting up an IoT scenario, one major step is the so-called onboarding. This term refers to the process of establishing a relationship between a physical entity and its virtual counterpart known as the digital twin. The physical entities that can be onboarded comprise the following:

- Persons
- Organizations
- Things

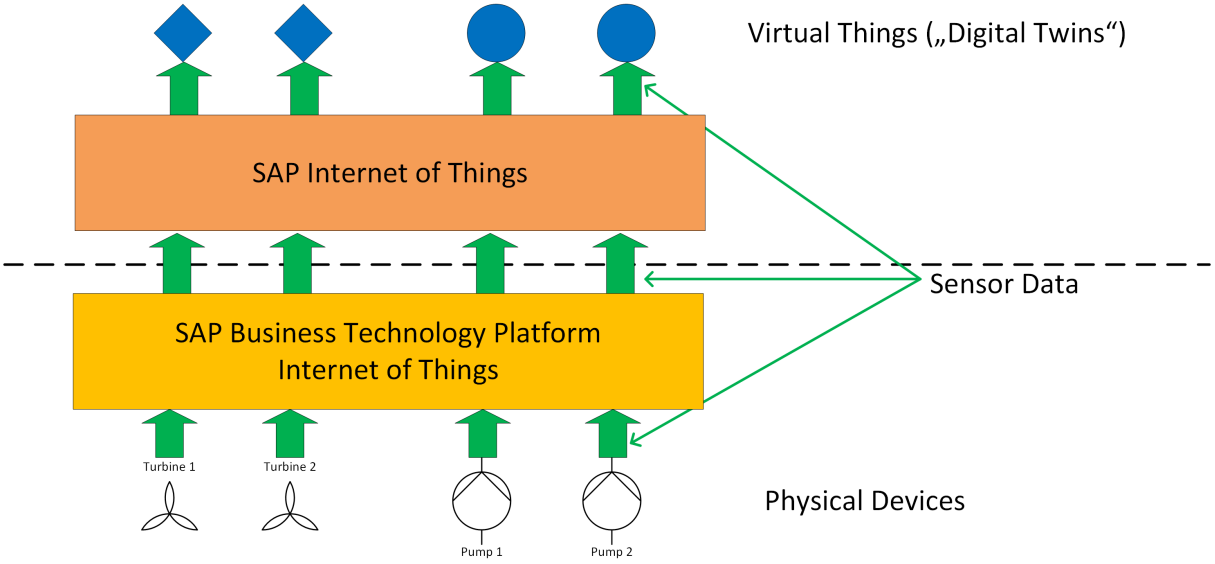
While onboarding of organizations and persons is covered by the Tenant Administration apps, the onboarding of things is a task that you perform with the Thing Modeler app. The onboarding process with the Tenant Administration apps basically means setting up a virtual mirror object for a real-world entity. In contrast to that, onboarding a thing does not only set up such a mirror object. Rather, a technical connection between a physical device and its digital twin is established. This connection is then used for transferring sensor data from a device into the IoT storage system for monitoring and analytics.

Onboarding Steps

Onboarding of a thing consists of the following activities:

- Identifying the physical devices to be covered in an IoT scenario.
- For each type of device: Defining a thing type with a property set that reflects the different kinds of sensor data that the device can provide. During thing type definition, the system automatically scans the device model and tries to match the sensors found in the device model with the properties that belong to the thing type.
- Create a device model, that is, a software layer (typically provided by SAP IoT services for SAP BTP) that mirrors the sensor types and sensors that are part of a physical device.
- For each device: Define a thing based on the previously defined thing types.

Here is a diagram showing the various object levels and software layers involved:



Onboarding Strategies

Individual Setup

You can use either SAP IoT services for SAP BTP for the Cloud Foundry Environment or an individual, manually developed solution accessing the API of the time series store service for importing time series data into SAP IoT by using the data ingestion services provided by the IoT Application Enablement API. In that case, you have to make sure that the device management software creates the necessary payloads in a way that can be consumed by the IoT Application Enablement services. Especially the payload structures and the defined thing properties must be synchronized so that the Thing Modeler app can map the two levels to each other.

Standard Integration with SAP IoT services for SAP BTP for the Cloud Foundry Environment

A mapping is possible only if the properties of the thing type have types assigned other than the following ones:

- *Byte Array*
- *JSON*

Mapping of Data Types Provided by SAP IoT services for SAP BTP for the Cloud Foundry Environment to SAP IoT Data Types

SAP IoT has no immediate access to physical devices. Rather, it relies on the SAP IoT services for SAP BTP layer that is in charge of providing a unified access to sensor-enabled hardware of all kinds. However, the type system of SAP IoT services for SAP BTP for the Cloud Foundry Environment is slightly different from the one that has been implemented for SAP IoT. For this reason, the data types of both layers must be mapped. The

mapping itself is taken care of by the system, so there is no need for manual activities here. The following table explains which types correspond to each other:

SAP IoT	SAP IoT services for SAP BTP for the Cloud Foundry Environment	Comment
Numeric	integer	
Numeric(s,n)	double, float	May lead to a loss of precision if the received value does not fit it the specified scale and precision
NumericFlexible	double, float	May lead to a loss of precision if the received value does not fit it the specified scale and precision
String	string	
Boolean	Boolean	
Date	date, long	
DateTime	date, long	
Timestamp	date, long	
JSON	Not supported	
LargeString	string	
ByteArray	Not supported	
Not supported	binary	
ThingID	string	
BusinessPartnerID	string	
GeoJSON	string	

Updating of Assigned Devices

If, on the SAP IoT services for SAP BTP layer, the settings are changed for a device that has already been associated with a thing, the consequences for data ingestion on the SAP IoT layer depend on which particular element of the device model is changed:

Consequences of Changes to Assigned Devices

Changed Element	Consequences
Device	Sensor-thing assignment is **not** affected.
Capability	If the name or the data type of a used property is modified, all affected sensor-thing assignments are deleted.
Sensor type	If the assigned capabilities are modified, all affected sensor-thing assignments are deleted.
Sensor	If the reference of a sensor to a sensor type is modified, all affected sensor-thing assignments are deleted.

In all of the cases listed above where the sensor-thing assignment is deleted, you have to reconnect thing and device manually before data ingestion can continue. It may also be advisable to consider how to handle

potential data gaps that may occur in the timespan during which a thing is disconnected from its associated device. Possible measures may comprise interpolation or randomization of missing values as well as cutting off the old time series and setting up a new one.

Related Information

Requirements for Successful Thing Onboarding [page 24]

3.1.2 Requirements for Successful Thing Onboarding

Prerequisites for a smooth interaction of SAP IoT and SAP IoT services for SAP BTP

Prerequisites and Requirements

During the onboarding of things, two different software layers are involved, SAP IoT and SAP IoT services for SAP BTP. While SAP IoT concentrates on the virtual aspects of an IoT scenario, SAP IoT services for SAP BTP takes care of the physical devices and sensors. For this division of labor to work smoothly, it is necessary to make sure that both software layers fit well together. To accomplish this, certain prerequisites have to be fulfilled.

> **i** Note
>
> For successful thing onboarding, the following prerequisites and technical limitations apply:
>
> - Although the device model offers the possibility to model a m:n (many to many) relationship between sensor types and capabilities, this is not supported for the integration with IoT Application Enablement. It is only possible to model a 1:n (one to many) relationship between sensor type and capability.
> - Capability names must be unique within a tenant
>
> > **i** Note
> >
> > There is one exception to this rule: If a device does not only support capabilities for reading measuring values, but also for setting a target value (for example, the temperature that is both measured and set by a thermostat), then the two capabilities must have the same name. That is, in this case, it is not only allowed, but **required** to use the same name for the two related capabilities. As a consequence, this leads to a new limitation: In the same tenant, a particular capability name can exist either once or twice, but not more.
>
> - The following data types are not allowed for thing properties:
> - Byte Array
> - JSON
> - The sensors of a physical device may send data at a maximum frequency of 1 Hertz (that is, one record per second or less).
> - Although the sensors can record data with a resolution of up to 1000 Hertz, they should send the data in batches of max. 0.1 Hertz frequency (that is, one batch every 10 seconds).

- The Centralized Data Model feature provided by SAP IoT services for SAP BTP cannot be used with SAP IoT.

Related Information

Create a Thing [page 27]
Thing Onboarding [page 21]

3.2 Maintain a Thing Type

Process steps for maintaining thing types

Context

As an administrator, you want to modify an existing thing type.

> **i** Note
>
> You should keep in mind the following aspects:
>
> - You can maintain thing types only in packages where you have write permission. Packages where this is **not** the case (that is, you are only authorized to display existing thing types) are flagged with a 🔒 symbol.
> - You cannot change the name of a thing type. This is because the name is used as a unique technical identifier by the system.
> - If things have already been derived from a thing type, you cannot change the type. The only exception is the image that has been associated with a type, which you can still change. However, changing the type image has no effect on things that have already been derived from the type, that is, thing images remain unchanged.

Procedure

1. From the launchpad, start the *Thing Modeler* app.

   The system presents the list of thing types available in the preselected package and loads the details of the first list entry into the details area. If you want to maintain a thing type in a different package, choose the package from the *Package* dropdown list.
2. From the list of thing types, choose the one you want to maintain. If you don't see the type that you want to change, use the *Search* to narrow down the number of types presented in the list.

3. Modify the field values or the assigned property sets according to your requirements.

   In case of a complex thing type that consists of a high number of property sets with many elements, you may find it handy to make use of the incremental *Search* field on top of the lists of elements in the detail area. Just start typing parts of the name you are looking for to narrow down the number of list entries.
4. You may also want to modify the existing mappings between sensor types and thing type properties on the *Connectivity* tab. However, if an assignment between the thing instance and the physical device already exists for the mapping, you cannot update the mapping by deleting or editing the already existing capability, properties and property set, properties. Instead, you can only update the mapping as follows:
   - Map a new capability property of the sensor device to a property of the property set
   - Map a new device capability to the property set
5. Choose *Save*.

   The *Save* button remains inactive as long as the data shown for the type is identical to the data that is already stored in the database.

3.3 Delete a Thing Type

Process steps for deleting a thing type from the system

Context

As an administrator, you want to delete a thing type from the system. Deleting a thing type can be useful in different situations, for example:

- The thing type is not referenced by any thing. In fact, this is a prerequisite for deleting a thing type.
- The thing type reflects the structure of a physical asset that is no longer in use in your company.

Procedure

1. From the launchpad, start the *Thing Modeler* app.

   The system presents the list of thing types available in the preselected package and loads the details of the first list entry into the details area. If you want to delete a thing type in a different package, choose the package from the *Package* dropdown list.
2. From the list of thing types, choose the one you want to delete. If you don't see the type that you want to delete, use the *Search* to narrow down the number of types presented in the list.
3. Make sure the currently displayed thing type is in fact the one that you want to delete. Once you are sure, choose *Delete*.

   The system presents a confirmation dialog where you have to confirm your decision to delete the type.
4. Confirm your decision.

Results

The thing type is deleted from the database. The system takes you back to the entry screen with the list of thing types.

3.4 Create a Thing

Process steps for creating a thing.

Context

As an administrator, you want to define a thing that shall act as a digital twin of a corresponding asset in the real world of your business.

> **i** Note
>
> The data provider for a new thing is a setting that you typically maintain upon creation. However, you are free to only define the basic settings of a new thing and wait with the data ingestion settings until everything else has been prepared for that. You can then complete the thing definition with the settings that you can make on the *Connectivity* tab.
>
> Although the individual settings can't be changed afterwards, you can still disconnect a thing from its device. In other words, should you realize that something has gone wrong with the data provider settings, you have to disconnect the thing from the device and reconnect it with the correct settings.
>
> Also, if, on the SAP IoT services for SAP BTP layer, the settings are changed for a device that has already been associated with a thing, the thing-device connection is lost, so that the thing cannot receive time series data anymore from the device. Again, thing and device have to be reconnected in such a case before data ingestion can continue.
>
> You can also maintain location coordinates for a thing. However, this data can only be maintained for an existing thing that has already been stored in the database. For more information, see Maintain a Thing [page 30].

Procedure

1. From the launchpad, start the *Thing Modeler* app.

   The system presents the list of thing types available in the preselected package and loads the details of the first list entry into the details area. If you want to create a thing for a thing type in a different package, choose the package from the *Package* dropdown list.
2. From the list of thing types, choose the one you want to use as the blueprint for the new thing. If you don't see the type that you want to change, use the *Search* to narrow down the number of types presented in the list.

Alternatively, you can also choose + *Add* at the bottom of the thing list. In that case, a dialog box opens where you can start defining the new thing right away (including the reference to the desired thing type).

3. Once you have selected the thing type, the system loads its data into the details area.
4. In the header area, choose *New Thing*.

   The system displays a dialog box where you can enter the details of the new thing:

   - *Name*: Technical name of the new thing.

     > **i Note**
     >
     > The name must consist of letters and numbers in the range of ["a..z", "A..Z", "0..9"] or the underscore "_" sign.

   - *Alternate Name*: Optional name field. You may use this field for maintaining thing names that follow a naming convention that can't be realized with the *Name* field.
   - *Description*: Short description of the thing in natural language.
   - *Thing Type*: Technical name of the thing type from which the new thing is derived. If you have started the creation from the thing type, that thing type is automatically entered into this field and can't be changed.

     > **i Note**
     >
     > With the thing type, all of the property sets that have been defined for the type are made available for the new thing.

   - *Authorization Group*: Technical name of the authorization group that is granting access to objects of the referenced thing type.
5. On the *Connectivity* tab, start by choosing the desired *Provider*. You will typically choose SAP IoT services for SAP BTP for the Cloud Foundry Environment here.
6. In the *Sensor/Mapping Selection* section, choose from the *Mappings* that have been defined for the associated thing type.
7. From the list of sensor types that are involved in the previously selected mapping, choose the one you want to use for the thing.
8. In the *Thing Model or Device Model Mapping* table, expand the property set entries so that you can see the individual properties of the thing.
9. In the *Device Property* column, choose from the list of matching properties the one that shall be assigned to the thing property.
10. Repeat the previous steps for every thing property that you want to have a device property assigned.
11. Choose *Save* to save the new thing.

    The System stores the new thing in the database, and the reference counter of the referenced thing type is incremented.

    In addition, the system informs you about the automatically generated ID. You can retrieve the generated ID later on by clicking the *Connection Info* symbol in the header area of the thing data.

    After the thing has been successfully created, you can also add an image of type PNG or JPEG to the thing, which might help you to easier identify the thing in the list of things presented by the Thing Modeler.

    > **i Note**
    >
    > Please be aware that an image that you assign to a thing type or a thing may contain any kind of visual information. There is **no** mechanism in place to check the image content in an automated fashion. It is

therefore up to you to ensure that all images uploaded to the system do comply with legal requirements and do not unveil any confidential information.

Results

The new thing has been created and onboarded, that is, it has been set up as a digital twin of a physical device, and it has been connected to that device. The onboarding result for a thing is also reflected by two closely related elements on the UI:

Onboarding Status of a Thing

Status Indicator	Action Link	Comment
Connected	*Disconnect*	Thing has been successfully onboarded and connected to device sensors. To change the sensor settings, first disconnect the thing, then reconnect it.
Disconnected	*Connect*	Thing hasn't yet been onboarded. Choose *Connect* to establish a relation between a thing and a device with sensors.

Now that you are done with defining a thing type or thing, let's have a look at the list of properties that you have assigned to it: The properties are presented in a hierarchical list that consists of either two or three levels, depending on the optional settings that have been made for a property in the *Thing Properties Catalog*:

Level	Object	Description
1	Property Set	An object that serves as a container for a set of individual properties. There can be one or more property sets assigned to a thing type or thing.
2	Property	The properties representing master data (*Basic Data*), sensor data (*Measured Values*) or data derived from sensor data (*Calculated Values*).
3	Reference Property (optional)	This level is only relevant for properties where addditional optional settings have been defined in the *Thing Properties Catalog*, such as thresholds, target value, or a unit of measure. If these additional settings have been defined, the system reflects that by automatically generating a reference property set in the backend. The reference properties are accessible via a respective API service method, which enables you to support scenarios where timeseries data for the same property is collected in different units of measure for different things. You can then use the reference property to recalculate the values so that they all refer to the same unit of measure and can be compared and analyzed in a useful manner (see Create a Property Set [page 34] for an example). If reference properties have been generated, the Thing Modeler displays them in the list of properties on level three. For each reference property, you can find its technical name that you need to know in order to access it via the API. For more information, see Reference Property Set Type.

Related Information

Thing Onboarding [page 21]
Requirements for Successful Thing Onboarding [page 24]

3.5 Maintain a Thing

Process steps for maintaining a thing

Context

As an administrator, you want to modify an existing thing.

> **i** Note
>
> You should keep in mind the following aspects:
>
> - You can maintain things only in packages where you have write permission. Packages where this is **not** the case (that is, you are only authorized to display existing things) are flagged with a 🕶 symbol.
> - You cannot change the name of a thing. This is because the name is used as a unique technical identifier by the system.
> - You cannot change the properties of a thing. This is because the properties are inherited from the thing type from which the thing has been derived upon creation. If you are not satisfied with the set of properties that are available for a thing, you should consider setting up a new thing type with the properties you would like to see.
> - You can change the connection between a thing and a physical device by changing the assignment using the available mappings that are offered on the *Connectivity* tab.

Procedure

1. From the launchpad, start the *Thing Modeler* app.

   The system presents the list of thing types available in the preselected package and loads the details of the first list entry into the details area. If you want to maintain a thing in a different package, choose the package from the *Package* dropdown list.
2. Choose the *Things* tab on top of the list of thing types and things.

   The system displays the list of things.
3. From the list of things, choose the one you want to maintain. If you don't see the thing that you want to change, use the *Search* to narrow down the number of things presented in the list.

4. Modify the field values according to your requirements. The following fields are only offered for maintaining an already existing thing (that is, the fields are **not** offered during thing creation): *Location (Latitude, Longitude)*.

   Location (Latitude, Longitude): Geographical coordinates of the place where the physical device is located.

   > **i** Note
   >
   > The two fields take values as decimal degrees in a range from -90..0..90 (*Latitude*) or -180..0..180 (*Longitude*), respectively. Negative values refer to the Southern hemisphere (*Latitude*) or to the Western longitudes, seen from the Greenwich Prime Meridian. Positive values refer to the Northern hemisphere (*Latitude*) or to the Eastern longitudes, seen from the Greenwich Prime Meridian.
   >
   > Entering location data for a thing is optional. However, if you decide to enter values into the *Location* fields, the system performs a validity check and prevents you from saving the thing in case of invalid entries.

   > Example
   >
   > The city of Brussels (Belgium) is located approximately at a latitude of 51° N and at a longitude of 4° E, resulting in a *Location* value pair of `51` and `4`. The city of Buenos Aires (Argentina) is located approximately at a latitude of 34° S and at a longitude of 58° W, resulting in a *Location* value pair of `-34` and `-58`.

   Note that the only fields that you can change for a thing - except for the location - are its *Description* and the assigned *Authorization Group*. In addition, you can attach a new image to a thing, or remove an existing image.

5. Choose *Save*.

   The *Save* button remains inactive as long as the data shown for the thing is identical to the data that is already stored in the database.

3.6 Delete a Thing

Process steps for deleting a thing from the system

Context

As an administrator, you want to delete a thing from the system. Deleting a thing can be useful in different situations, for example:

- You have found an issue with the properties of the thing type that is referenced by the thing. for correcting the thing type, you first have to remove all things that refer to the type.
- The thing represents a physical asset that is no longer in use in your company.

> **i Note**
>
> Before you delete a thing, you should double-check if the thing is really no longer needed. Accidentally deleting a thing might lead to data loss or incomplete time series data.

Procedure

1. From the launchpad, start the *Thing Modeler* app.

   The system presents the list of things available in the preselected package and loads the details of the first list entry into the details area. If you want to delete a thing in a different package, choose the package from the *Package* dropdown list.
2. Choose the *Things* tab on top of the list of thing types and things.

   The system displays the list of things.
3. From the list of things, choose the one you want to delete. If you don't see the thing that you want to delete, use the *Search* to narrow down the number of things presented in the list.
4. Make sure the currently displayed thing is in fact the one that you want to delete. Once you are sure, choose *Delete*.

   The system presents a confirmation dialog where you have to confirm your decision to delete the thing.
5. Confirm your decision.

Results

The thing is deleted from the database. The system takes you back to the entry screen with the list of thing types.

4 Thing Properties Catalog: Overview

Overview of the Thing Properties Catalog App

Introduction

When you define thing types and things with the Thing Modeler app, you will notice soon that that app is focusing on things and thing types **as a whole**. That is, the available things and thing types are listed and you can perform certain global activities on them (such as deriving a thing from a type, copying, deleting, setting an icon, and so on) but you cannot directly define the inner structure of a thing in terms of its properties or the measured values that can be collected for it.

This kind of technical properties is what you define in the Thing Properties Catalog, and once the relevant property sets are available, you can assign them by reference to one or more thing types in the Thing Modeler. Here is a high-level description of how to proceed:

You have a clear understanding of the technical details of each of your physical assets (for example, because you have collected all the technical data sheets for your machinery and want to start with that). In that case, you start with the Thing Properties Catalog and define the sets of properties that are specific for a particular thing type. Once done, you switch to the Thing Modeler app, create the thing types, assign the respective property sets to them and finally derive the things from the thing types.

Property Set Types

When you create a new property set (that is, a set of properties that you want to assign as a whole to a particular thing type), you must define the type of data that is specific for that type of property set. The following property set types are available:

- *Basic Data*: You use this property set type for master data of a thing that remain stable over time under controlled production conditions (for example, length, height, maximum power consumption, noise emission, number of extension slots, and so on).
- *Measured Values*:
  This property set type is meant for mass data that is sent continuously to the system by the sensors that are attached to a particular thing (for example, water temperature, oil pressure, energy consumption). As opposed to basic data, measured values are expected to vary all the time depending on the environment conditions to which a thing is currently exposed, like climate data, workload, current state of material wear, and so on.
- *Calculated Values*: This property set type is based on measured values, which are not displayed as individual raw values but as a result of an aggregation, or calculation, that the system has performed in advance. A typical example of this is an average value based on a certain time frame like "moving average temperature over the last two hours".

> **i** Note
>
> The *Measured Values* as well as the *Calculated Values* property set types are also used for presenting status values for a thing. Status values often have a relationship to a value that has been measured or calculated.

They are transmitted when a certain event has occurred for a thing (for example, "maximum temperature exceeded", "material supply exhausted", "safety guard has been removed", and so on)

! Restriction

You can define the property set type only upon creation of a new property set. It cannot be changed anymore after you have saved the property set.

4.1 Create a Property Set

Process steps for creating a property set and related objects.

Context

As an administrator, you want to define a property set (that is, a set of properties that belong together) that is related to a particular thing type.

✤ Example

You want to set up a hierarchy of thing types referring to different kinds of vehicles with the relevant properties assigned to the appropriate levels of the thing type hierarchy. You may start with an abstract thing type "Vehicle", with "Land Vehicle" and "Water Vehicle" as its descendants. From "Land Vehicle", you derive "Wheel-based Vehicle" and "Chain-based Vehicle". Finally, for the "Wheel-based Vehicle", you set up a property set "Wheels", with the following set of basic properties (of course, many more may be needed in a real-world scenario):

- front left size
- front left pressure/specified
- front left pressure/actual
- front right size
- front right pressure/specified
- front right pressure/actual
- rear left size
- rear left pressure/specified
- rear left pressure/actual
- rear right size
- rear right pressure/specified
- rear right pressure/actual
- drive type [front | rear | four wheel]

Once you are done with the property set, you assign it to the "Wheel-based Vehicle" thing type and then go on deriving further refinements of this kind of vehicle, such as "Car", "Truck", "Bus", and so on, which all inherit the properties you have defined for the "Wheels" property set.

Procedure

1. From the launchpad, start the *Thing Properties Catalog* app.

   The system presents the list of property sets available in the preselected package and loads the details of the first list entry into the details area. If you want to create a property set in a different package, choose the package from the *Package* dropdown list.

2. At the bottom of the property set list, choose + *Add*.

   The system displays the *Create Property Set* dialog box.

3. Enter a unique technical *Name* for the new property set (required) and a short description of the property set's purpose.

   For more information on the naming convention, see Create a Property Set Type chapter in API Reference Guide.

   Note that when you access the property set in the Thing Properties Catalog app, the URL displays the property set name as `<package Name>/<data category>/<property set name>`. For example, `sap.iot.core.automobiles/MasterData/CarInfo`.

4. Decide which of the available property set categories you want to assign to the new property set.

   Note that you cannot change this setting anymore after the property set has been created. For more information, see Thing Properties Catalog: Overview [page 33].

5. Once you are done, choose *Save*.

   The system presents the details screen for the new property set. The screen looks different, depending on which type you have chosen:

   - *Basic Data*: The system displays an empty list of properties where you can add properties whose values are supposed to be stable over time (like length, width, weight, voltage, and so on).
   - *Measured Values*: You can define properties that represent values that have been determined by sensors that are attached to a thing.
   - *Calculated Values*: You can define properties that represent values that have been determined by sensors and have been aggregated and calculated by a preprocessing engine in the backend.

   > **i Note**
   >
   > For both measured values as well as calculated values you can also define status values that may - but do not have to - correspond to the measured or calculated values that you define in the scope of a particular property set.
   >
   > For measured values of type *String*, *Thing ID*, or *Business Partner ID*, you can specify whether the incoming values shall be treated as measures (which is the default) or as a *Dimension*. With that, you have the following options:
   > - For numeric time series values, the system automatically creates aggregates based on different time horizons (for example, average value over the last minute, day, or week).
   > - In case of a measured value that is flagged as a dimension, the values can be used for setting up additional aggregates that you can combine with the time-based aggregates. For example, in a production scenario, a machine producing certain parts might provide a value that indicates the product line to which each produced part belongs. Defining this product line property as a dimension lets you narrow down the results of all kinds of analytical processing according to this additional dimension (for example, the maximum pressure used to produce a part broken down by product line).

For measured values of type *Integer*, *Decimal*, *Float*, or *String* you can specify whether the system shall store an additional *Quality Code* that accompanies each recorded value. The quality code is a numeric value that describes the reliability (for example, regular value versus outlier) of a particular measured value. For example, when you collect time series data at a fixed interval, it may happen that for several reasons a measured value is lost. You can decide that you still want, or need, to use the time series data by replacing the missing values by interpolated values that you calculate to fill the gaps. However, that results in a lower rating expressed by the quality code. Note that the system does only calculate aggregates for measurements that reach a quality assessment of "good" according to the quality code definitions proposed by the Open Platform Communications (OPC) specification. For more information, see OPC Quality Codes .

For measured values of type *Integer*, *Decimal*, and *Float*, you can specify whether the values shall be actually treated as measured values (as the name suggests) that are received from the connected sensors or if these values represent a desired *Target Value* for the property in question. From a technical perspective, this means that the data flow direction is reversed here: Target values are **sent to** the underlying device model, whereas measured values are **received from** the device model. It is then up to the system to take action if a measured value from a particular sensor does not match the corresponding target value.

For all types of measured values, you can assign a Unit of Measure. You can enter a free text (up to three characters) depicting the unit of measure or select one of the predefined values from the drop-down list. This value appears in the **Unit of Measure** column when you view the properties of a thing in Thing Modeler app. To know the complete list of predefined unit of measure values, you can use the Value Helps services. For the *Unit of Measure* already assigned, you can decide whether overwriting the predefined unit of measure with another one shall be allowed. Activating this option may sound weird and a danger to consistency at first glance. However, the feature allows you to cover business scenarios like the one in the following example:

> ### Example
>
> For a global company with subsidiaries in Europe and the United States, you want to implement a car fleet monitor, which, among others, keeps you informed about the average gas consumption and the maximum speed reached by each vehicle. However, you soon realize that you have to adapt the way how the respective values are treated to what is typical of the different cultures at the various locations: In Europe, speed is measured in kilometers per hour (km/h), whereas in the US, miles per hour (mph) is used. For gas consumption, things are even more complicated because there are not only different units of measure used, but the overall concept of thinking about consumption differs: Americans think about how far they can drive with a given volume of gas (miles per gallon, mpg), while Europeans think about how much gas is consumed when driving a certain distance (liters per 100 km, l/100 km).

The example makes it clear that this scenario can only be solved by either forcing employees to behave in a way that they perceive as unnatural for their cultural background, or by making the system smarter in terms of deferred data transformations and recalculations. The second approach is what you can accomplish by setting the *UoM Overwritable* flag for a property. Note, however, that this setting has no further consequences for the thing modeling process with the Thing Modeler apps. Instead, you need to reflect this setting by making use of the API service for accessing reference properties. For more information, see Create Data for Reference Properties . In addition, when you create a thing, you can enter values for the reference properties, such as Unit Of Measure.

6. In the *Properties* section or in one of the *Values* sections in the detail area, choose + *Add* to create a new property or value.

The system displays a dialog box where you can enter the details for the new property or value.

> **i Note**
>
> For properties that you want to use for rule processing, you need to adhere to some naming conventions that are validated by the rule processing framework. For more information, see Naming Conventions for Rule Processing.

7. Once you are done, choose *Save*.

   The new property or value is added to the property set. You can continue adding further elements to the property set by repeating the previous steps.

8. Once you are finished with adding elements to the property set, choose *Save* at the bottom of the screen to save the new property set.

> **i Note**
>
> Now that you are done with setting up the new property set, we recommend checking whether this property set might be used for storing data that could be classified as personal data, or even critical personal data. Here are some examples:
>
> - A sensor determining geolocation data is attached to a device that a person carries around throughout the day.
> - A medical device collects health-related data such as body temperature or blood pressure of a person.
>
> If you come to the decision that any of the properties of the property set fulfills the criteria for personal data, or critical data, you must indicate this by choosing the corresponding *Sensitivity Level*. Any sensitivity level other than `None` prevents a property set from being assigned to a rule vocabulary defined for rule processing. Also, if the sensitivity level of an initially non-classified property set is elevated after that property set has already been assigned to a rule vocabulary, all referencing rules are automatically deactivated and cannot be executed anymore.

You can add or delete elements to or from a property set at a later point in time whenever you want, even if the property set has already been assigned to a thing type.

Results

The new property set is stored in the database and can be assigned to one or more thing types in the *Thing Modeler*.

> **i Note**
>
> For the recommended maximum number of properties in a property set, see the *Limitations* section in Modeling Things: Overview [page 3].
>
> For a complete list of all property data types supported by SAP IoT as well as their counterparts in SAP IoT services for SAP BTP for the Cloud Foundry Environment, see Thing Onboarding [page 21].

Related Information

Additional Settings for Numeric Values [page 38]
Create a Vocabulary

4.1.1 Additional Settings for Numeric Values

Additional settings for properties with numeric values

Introduction

When you define a property in the Thing Properties Catalog, you will soon notice that the attributes for properties are basically the same: They all have a name, a description, a type, and so on. However, the situation looks different when you define the data type for a measured value or a calculated value as numeric (integer, decimal number, or float).

Defining the Length of Numeric Values

For numeric properties that are not of type integer, you have to define the length of both the integer as well as the fractional part. Here, again, the system makes a difference between the subtypes decimal number and float: For decimal numbers, defining the number of digits is mandatory, whereas for float, it is optional. This may seem inconsistent at first glance, but there is good reason for this difference in system behavior because both types serve different purposes:

- Decimal numbers can be used to display numbers in a visually unified format (for example, displaying the calculated average of measured values from a sensor with a user-defined, fixed number of digits for the integer as well as for the fractional part). Such visual unification may, on the other hand, lead to a loss of precision.
- Floating point numbers are defined for maximum precision. For this numeric type, the system automatically determines the number of digits that are needed to correctly display a value. On the other hand, listing many floating point numbers in one column typically leads to an unwanted visual appearance. To avoid such disarrangement, you can define (but you don't have to) a field length of your choice for this kind of data.

Thresholds

For all numeric properties (that is, integer, decimal number, and float), you have the option to define up to four different thresholds:

- Lowermost

- Lower
- Upper
- Uppermost

During the property definition, all you have to do is to decide whether you want threshold support or not. To accomplish this, you may activate any combination of thresholds out of the set of four thresholds and specify a custom label for the individual thresholds. However, this is only a preparatory step for a threshold definition.

The second step is following a bit later in the thing modeling process, when it comes to setting up a concrete thing. That is, the following steps have been done already:

1. Define a property set with threshold-enabled properties.
2. Define a thing type.
3. Assign the property set to the thing type.
4. Derive a concrete thing from the thing type.

At this stage, you will notice that for the thing, not only a field for the current measured value or calculated value is displayed, but also additional fields for threshold values, one field for each threshold that you have activated for the respective property.

> **i Note**
>
> It may be worth mentioning that these additional fields become visible in the Thing Modeler app only after you have manually expanded the respective tree structure in the table of properties.

Now, in the Thing Modeler, you specify the values that shall be treated as thresholds for the property in question.

> **Example**
>
> You want to monitor the tire pressure for the tires that are mounted on the wheels of a vehicle. When the sensors determine a critical value, the system shall alert you with the help of some warning mechanism that you want to implement. To accomplish this, proceed as follows:
>
> 1. Define a property set that contains a property with unit of measure = "bar" for the tire pressure with all four thresholds enabled.
> 2. Assign that property set to a "Vehicles" thing type.
> 3. Derive a "Car" thing from the thing type.
> 4. In the *Measured Values* or *Calculated Values* section, expand the property for which you have enabled the thresholds. Enter the following values:
>   - *Lowermost*: `1.4`
>   - *Lower*: `2.0`
>   - *Upper*: `2.7`
>   - *Uppermost*: `2.9`
>
>   > **i Note**
>   >
>   > These thresholds should be appropriate for a tire that is specified for a pressure between 2.2 and 2.5 bar, depending on the boundary conditions during operation (such as load, temperature, speed, and so on).
>
> 5. Make sure your monitoring application takes these threshold values into account by permanently comparing the current actual value retrieved from the pressure sensor against the thresholds.

Whenever a threshold is met or exceeded by the current value, you let the system alert the operator, for example, with a set of error messages like this:

- *Lowermost*: "Danger: Insufficient tire pressure! Stop the car!"
- *Lower*: "Warning: Tire pressure too low. Check the pressure."
- *Upper*: "Warning: Tire pressure too high. Check the pressure."
- *Uppermost*: "Danger: Tire pressure exceeds the specified maximum. Stop the car!"

4.2 Maintain a Property Set

Process steps for maintaining property sets

Context

As an administrator, you want to modify an existing property set.

i Note

You should keep in mind the following aspects:

- You cannot change the data type of an existing property set (basic data, measured or calculated values). This setting can only be made upon creation of a property set.
- You cannot change the name of a property set. This is because the name is used as a unique technical identifier by the system.
- If a property set has already been assigned to a thing type, you cannot delete it.
- If a property set contains a measured value that has been flagged as a dimension, this setting cannot be changed. For more information on the *Dimension* setting, see Create a Property Set [page 34].
- If a property set contains measured value or calculated value with threshold values assigned, you cannot change the custom label of the threshold. You must delete the assigned threshold and reassign the threshold with a different custom label.
- If you elevate the *Sensitivity Level* of an initially non-classified property set after that property set has already been assigned to a rule vocabulary, all referencing rules are automatically deactivated and cannot be executed anymore.
- Be very careful before you delete an existing property or value from a property set that is already used by things via their assigned thing type. Removing such elements could lead to unwanted gaps in existing time series data that has been collected in the past for the element you want to delete. The same applies for changing the data type of a property set element.

Procedure

1. From the launchpad, start the *Thing Properties Catalog* app.

The system presents the list of property sets available in the preselected package and loads the details of the first list entry into the details area. If you want to maintain a property set in a different package, choose the package from the *Package* dropdown list.

2. From the list of property sets, choose the one you want to maintain. If you don't see the property set that you want to change, use the *Search* or the *Filter* to narrow down the number of property sets presented in the list.
3. Modify the field values or the properties according to your requirements.

   In case of a complex property set that consists of a high number of elements, you may find it handy to make use of the incremental *Search* field on top of the lists of elements in the detail area. Just start typing parts of the name you are looking for to narrow down the number of list entries.
4. Choose *Save*.

   The *Save* button remains inactive as long as the data shown for the property set is identical to the data that is already stored in the database.

Related Information

Create a Vocabulary

4.3 Delete a Property Set

Process steps for deleting a property set from the system

Context

As an administrator, you want to delete a property set from the system. Deleting a property set can be useful in different situations, for example:

- The property set has been assigned to a thing type that is not referenced by any thing.
- The property set has been assigned to a thing type that reflects the structure of a physical asset that is no longer in use in your company.
- The property set has been defined in a too specific way so that it can be used for only a few things. You want to replace it by a set of less complex property sets that can be combined more easily, thereby increasing the reuse potential of your thing model, which in turn improves consistency.

> **i** Note
>
> You should keep in mind the following aspects:
>
> - If a property set has already been assigned to a thing type, you cannot delete it. If it is important for you to delete a property set from the system, you can accomplish this by removing all thing type assignments of the property set.

Procedure

1. From the launchpad, start the *Thing Properties Catalog* app.

   The system presents the list of property sets available in the preselected package and loads the details of the first list entry into the details area. If you want to delete a property set in a different package, choose the package from the *Package* dropdown list.
2. From the list of property sets, choose the one you want to delete. If you don't see the property set you are looking for, use the *Search* or the *Filter* to narrow down the number of property sets presented in the list.

   The property set definition is displayed in the details area.
3. Make sure the currently displayed property set is in fact the one that you want to delete. Once you are sure, choose *Delete*.

   The system presents a confirmation dialog where you have to confirm your decision to delete the property set.
4. Confirm your decision.

Results

The property set is deleted from the database. The system takes you back to the entry screen with the list of property sets.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP