

# SAP Business Network Global Track and Trace

## Guide for Model Administrators (Creating and Deploying Version 2 Models)

PUBLIC

Document Version: LBN 2.0 – 9 December 2023



# TABLE OF CONTENTS

1	INTRODUCTION .....	4
1.1	About This Document .....	4
1.2	Target Audience.....	4
1.3	Prerequisites .....	4
2	ABOUT GLOBAL TRACK AND TRACE.....	5
2.1	Basic Terms .....	5
2.2	Structure.....	7
3	MODEL OVERVIEW PAGE .....	8
3.1	In-app Help .....	8
3.2	Standard and User Models .....	9
3.3	The Model Card.....	9
3.4	Import Model .....	10
3.5	Export Model .....	10
3.6	Delete Model.....	10
4	CREATE AND DEPLOY A MODEL .....	11
4.1	Create a Model from Scratch.....	11
4.2	Edit an Existing Model .....	12
4.3	Deploy a Model .....	12
4.4	Model Header .....	13
4.5	Tracked Process .....	15
4.5.1	Create Tracked Process.....	15
4.5.2	Edit Tracked Process .....	16
4.5.3	Delete Tracked Process .....	16
4.5.4	Field List.....	17
4.5.5	Admissible Planned Events.....	19
4.5.6	Admissible Unplanned Events .....	21
4.6	Field Type .....	22
4.6.1	Create Field Type.....	22
4.6.2	Edit Field Type .....	23
4.6.3	Delete Field Type .....	23
4.7	Event Type.....	24
4.7.1	Create Event Type .....	24
4.7.2	Edit Event Type.....	26
4.7.3	Delete Event Type.....	26
4.8	Code List .....	27
4.8.1	Create Code List .....	27
4.8.2	Edit Code List.....	28
4.8.3	Delete Code List.....	28
4.9	IDOC Integration .....	29
4.9.1	To Setup IDOC Integration.....	30

<b>4.10</b>	<b>Visibility Provider Integration .....</b>	<b>31</b>
<b>4.10.1</b>	<b>To Setup Visibility Provider Integration .....</b>	<b>33</b>
<b>4.11</b>	<b>Planned Event Extension.....</b>	<b>34</b>
<b>4.11.1</b>	<b>Create Planned Event Extension Fields.....</b>	<b>34</b>
<b>4.11.2</b>	<b>Edit Planned Event Extension Fields .....</b>	<b>34</b>
<b>4.11.3</b>	<b>Delete Planned Event Extension Field .....</b>	<b>34</b>
<b>4.12</b>	<b>Event to Action (Business Rules).....</b>	<b>35</b>
<b>4.12.1</b>	<b>Add a Script to Your Model .....</b>	<b>35</b>
<b>4.13</b>	<b>Translation .....</b>	<b>36</b>
<b>4.13.1</b>	<b>Supported Language.....</b>	<b>36</b>
<b>4.13.2</b>	<b>Translation Dialog .....</b>	<b>36</b>
<b>5</b>	<b>DEPLOYED MODELS .....</b>	<b>37</b>
<b>5.1</b>	<b>Runtime View .....</b>	<b>37</b>
<b>5.2</b>	<b>Write Service .....</b>	<b>37</b>
<b>5.2.1</b>	<b>Trying Out API.....</b>	<b>37</b>
<b>5.3</b>	<b>Read Service .....</b>	<b>37</b>
<b>5.4</b>	<b>Deployed View .....</b>	<b>38</b>
<b>6</b>	<b>MORE TASKS.....</b>	<b>39</b>
<b>6.1</b>	<b>Activate/Deactivate the Model.....</b>	<b>39</b>
<b>6.2</b>	<b>Delete Business Data of the Model.....</b>	<b>39</b>
<b>6.3</b>	<b>Export Business Data of the Model .....</b>	<b>40</b>
<b>6.4</b>	<b>Enable/Disable the Model Cache .....</b>	<b>40</b>
<b>6.5</b>	<b>Enable/Disable Event to Action Log of the Model .....</b>	<b>41</b>
<b>6.6</b>	<b>View the Model History .....</b>	<b>41</b>
<b>7</b>	<b>CHECK PROCESSES AND EVENTS .....</b>	<b>42</b>
<b>7.1</b>	<b>About the CPE App .....</b>	<b>42</b>
<b>7.1.1</b>	<b>In-app Help .....</b>	<b>42</b>
<b>7.1.2</b>	<b>Filters .....</b>	<b>42</b>
<b>7.1.3</b>	<b>Views.....</b>	<b>43</b>
<b>7.1.4</b>	<b>Process List .....</b>	<b>43</b>
<b>7.1.5</b>	<b>Process Details.....</b>	<b>45</b>
<b>7.1.6</b>	<b>Planned Event Details .....</b>	<b>46</b>
<b>7.1.7</b>	<b>Event Details .....</b>	<b>46</b>
<b>7.2</b>	<b>Check Active Process Types .....</b>	<b>48</b>
<b>7.2.1</b>	<b>Check Active Processes.....</b>	<b>48</b>
<b>7.2.2</b>	<b>View Tracked Process Details.....</b>	<b>48</b>
<b>7.2.3</b>	<b>View Planned Event Details.....</b>	<b>49</b>
<b>7.2.4</b>	<b>View Actual Event Details.....</b>	<b>50</b>
<b>8</b>	<b>APPENDIX: EVENT TO ACTION SCRIPT.....</b>	<b>51</b>

## 1 INTRODUCTION

### 1.1 About This Document

This documentation describes how to perform all the tasks required to manage version 2 models for SAP Business Network Global Track and Trace. This involves using the following apps:

- Manage Models (MM) app
- Check Processes and Events (CPE) app

#### **Recommendation**

Before you start working your way through this document, ensure you have the most recent version of this document that is available from:

[help.sap.com/gtt](https://help.sap.com/gtt)

### 1.2 Target Audience

The target audience for this guide is Administrator – Data Models (GTT).

### 1.3 Prerequisites

You are able to log on to SAP Business Network Global Track and Trace as an Administrator – Data Models (GTT).

## 2 ABOUT GLOBAL TRACK AND TRACE

The main purpose of SAP Business Network Global Track and Trace is to track processes that are influenced by business events. All tracked processes and business events have a common core behavior and share a common set of basic attributes.

Therefore, the GTT solution consists of the following:

- A core engine that provides generic functionality to track processes, handle business events and provide the modeling environment described in this document.
- GTT apps that enrich the core engine's generic functionality by adding domain specific semantics in terms of data and behavior.

### 2.1 Basic Terms

The applications of SAP Business Network Global Track and Trace deal with the following central objects and terms:

#### Tracked Process

A tracked process comprises the following:

- Involved parties acting in or being affected by the process
- Track multiple other processes
- A set of planned events that are expected to happen within the tracked process.  
Note: Planned events are called "milestones" in SAP ERP applications like SAP Event Management.
- A set of events that actually happened against the process. This could be a realization of a planned event or also an unplanned event

#### Event

Events can be planned events or unplanned events that might occur during a process. When an event happens, it becomes an actual event. An actual event needs to be matched against a planned event of that process. If not, it becomes an unplanned event.

#### Planned Event

- Has a fixed semantic.
- Has a planned date and time (with time zone) + tolerance window, planned location, and other planned data.

Examples:

- goods issued (planned date 2016-07-06; planned location gate 1)
- goods received (planned date 2016-07-07; planned location packstation 801)

#### Unplanned Event

- Has a fixed semantic as well
- Is not planned, but can be foreseen that it occurs with a certain likelihood, but cannot be planned thus has no planned data
- Examples include: delay in delivery, damage of goods
- Or is a piece of information at an arbitrary point in time, such as the current position and temperature of a container

#### Actual Event

- Is an event that actually happened
- Is either matched against a planned event of a process and complements planned data with actual data or becomes an unplanned event of the process
- Triggers changes of attributes of a tracked process

## Timestamps

Every actual event has two timestamps:

1. Business timestamp: the time at which the event physically happens.
2. Technical timestamp: the time at which a user reports the event in the system.

Usually, the timestamps for the same event are different, with the time of the business timestamp before that of the technical.

The timestamp uses the UTC time.

The event status is determined by comparing the planned business timestamp with that of the actual.

Note: For simplicity in the following examples, only one planned business timestamp is used to determine the event status. In practice rather than a single planned business timestamp, a range is used from the planned business earliest timestamp to the planned business latest timestamp.

### Example 1

The times in the timestamps for an “arrival” event are as follows:

- Planned business timestamp: 2:00 pm
- Actual business timestamp: 2:30 pm
- Actual technical timestamp: 3:00 pm

The comparison of planned and actual business timestamps determines the event status:

- Planned business timestamp: 2:00 pm
- Actual business timestamp: 2:30 pm -> event status **LATE**

### Reporting an Event Again

The situation is more complicated if the same event is reported again, or even multiple times. Then the sequence of actual events is determined by the setting you made when you created or edited the tracked process for [Affect Planned Event Status By](#). You can select either: [Actual Business Timestamp](#) or [Actual Technical Timestamp](#). For more information, see [Create Tracked Process](#).

But in either case, the new timestamp must be later than the previous one, otherwise it is ignored.

### Example 2

As in the previous example, the times in the timestamps for an “arrival” event are as follows:

- Planned business timestamp: 2:00 pm
- Actual business timestamp 1: 2:30 pm -> event status LATE
- Actual technical timestamp 1: 3:00 pm

The event is reported for a second time with the following timestamps:

- Actual business timestamp 2: 1:00 pm
- Actual technical timestamp 2: 4:00 pm

If you set: [Affect Planned Event Status By](#) actual business timestamp, then:

- The actual business timestamp is used to determine the sequence of actual events.
- As the second actual business timestamp (1:00 pm) is before that of the first (2:30 pm), the second timestamp is ignored.
- So, the actual business timestamp remains as 2:30 pm and the event status remains **LATE**.

If you set: [Affect Planned Event Status By](#) actual technical timestamp, then:

- The actual technical timestamp is used to determine the sequence of actual events.
- As the second actual technical timestamp (4:00 pm) is after that of the first (3:00 pm), the second timestamp is valid and the actual technical timestamp becomes 4:00 pm.
- Event status is then determined by comparing the second event's actual business timestamp with that of the planned.
- As the actual business timestamp (1:00 pm) is before the planned business timestamp (2:00 pm), the event status becomes **EARLY**.

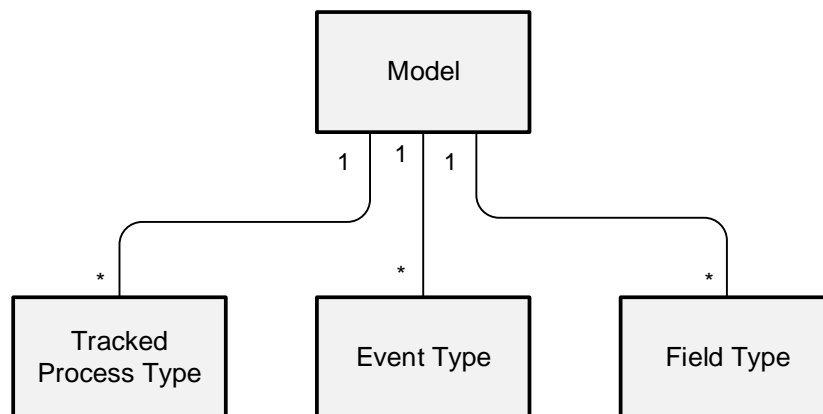
## 2.2 Structure

Different models can be created in the MM app. Each model must contain one or more tracked process types and event types. Each tracked process type and event type contain some pre-defined core model fields that you cannot change. You can add user-defined fields to each tracked process type and event type. For each tracked process type, some event types can be specified as either an:

- Admissible Planned Event type:  
specifies the event types that can be used in planned events of a tracked process with this process type or an
- Admissible Unplanned Event type:  
specifies the event types that can be used in unplanned events of a tracked process with this process type.

As shown in the following diagram, some other field types can be defined:

- Tracked process type
- Event type
- Field type



### 3 MODEL OVERVIEW PAGE

MM app starts by opening the *Model Overview* page. This lists up to 20 models. The model count is displayed at the bottom. For example: (20/100) indicates that 20 models are displayed of a total of 100. If there are more than 20 models, click the [More](#) button to see the next 20. The information on each model is displayed on a model card.

To find your model, you can simply check the list or search by typing its model name. To select your model, click its model card.

Also, on the *Model Overview* page, you can do the following:

- Create a model
- Import a model
- Export a model
- Delete a model
- Refresh the latest data of models on the page by clicking the Refresh icon.

On the top right, a [Sample Codes](#) menu is provided with direct links to download the sample codes. There are two options in the drop-down list:

- [For Template Apps](#): links to the following GTT repository in the SAP Samples organization on GitHub.com where you can download the sample codes of template apps: <https://github.com/SAP-samples/logistics-business-network-gtt-samples>
- [For Fulfillment Tracking Apps](#): links to the following GTT repository in the SAP Samples organization on GitHub.com where you can download the SAP ERP sample codes of Fulfillment Tracking apps: <https://github.com/SAP-samples/logistics-business-network-gtt-standardapps-samples>

#### 3.1 In-app Help

When you launch the MM app, you can turn on in-app help that provides on-screen explanations of key fields and areas on the screen. To turn on in-app help:

- On the top right on the screen, click the [?](#) button.
- the *Help Topics* Panel appears on the right

Once in-app help is turned on, you can:

- search the displayed help topics for text you type
- click the information icon to see help text on that topic
- hide the *Help Topics* Panel by clicking the *Hide (>>)* button on the bottom right
- toggle *Help* off



### 3.2 Standard and User Models

You have a choice of two types of model:

1. Standard models are predefined and delivered in the product.
2. User models are created by model administrators.

The standard model is consumed by the Fulfillment Tracking apps. It provides:

- standard tracked processes with model fields, planned and unplanned events in the [Tracked Process](#) tab
- standard field types with model fields in the [Field Type Pool](#) tab
- standard event types with model fields in the [Event Type Pool](#) tab
- standard code lists with model codes in the [Code List](#) tab
- standard IDOC mapping in the [IDOC Integration](#) tab
- standard visibility provider mapping in the [Visibility Provider Integration](#) tab
- standard planned event extensions in the [Planned Event Extension](#) tab
- standard declaration and script in the [Event to Action](#) tab.

You cannot delete a standard model. You can edit it to create user-defined tracked processes, field types, event types, code lists, planned event extensions, and event-to-action scripts. User-defined fields in the standard model also support IDOC mapping and visibility provider mapping.

Note that the name of user-defined tracked processes, field types, event types, code lists, and planned event extensions must start with the case-insensitive letters “ZZ”.

The standard model is integrated with [LBN code lists](#).

### 3.3 The Model Card

The information displayed on the model card includes:

- Name: in lower case with numerals except as the first character. For example, process17 or salesorder317. The name must be unique in the tenant.
- Draft version status:
  - **Blank**: the model's draft version is the same as the deployed version.
  - **Draft**: the model's draft version is different from the deployed version.
- Deployed version status:
  - **Blank**: the model has not been deployed.
  - **Active**: the deployed version is active — the write/read services of the model are running.
  - **Inactive**: the deployed version is inactive — the write/read services of the model are disabled.
- Description: up to 255 characters describing the model.
- Namespace: identifies the model and includes its name. This is unique across SAP Business Network for Logistics.
- Last Operation Status:
  - **Deployment Success**
  - **Deployment Warning**
  - **Deployment Pending**
  - **Deployment Error**
  - **Deletion Pending**
  - **Deletion Error**

The icon for each stage is color-coded, so you can see its status at a glance: green for success, red for error, and orange for warning and pending.
- [Event to Action Log](#):
  - **Off**: the Event to Action Log is disabled.
  - **On**: the Event to Action Log is enabled.

### 3.4 Import Model

#### Context

You can import a single model from a JSON file that is sized below 2MB.

#### Procedure

1. On the *Model Overview* page, click the [Import](#) button.
2. In the popup dialog, select the json file to upload by clicking the [Browse...](#) button. Once you have selected your file, click [Import](#).
3. When the file is uploaded, you see one of the following three kinds of messages:
  - **Error:** the message identifies what went wrong such as the file is broken, it cannot be decoded, or it has an incompatible data structure. It may also suggest what to do next.
  - **Warning:** when the name of the uploaded model already exists in the MM app, you must confirm whether to overwrite the existing model or abort the upload.
  - **Success:** your imported model was created successfully.

### 3.5 Export Model

#### Context

You can export a single model as a JSON file.

#### Procedure

1. On the *Model Overview* page, select the model you want to export, then click the [Export](#) button.
2. A popup dialog appears. If the model is already deployed, you can choose to export either the draft version or the deployed version. If the model is not deployed, you can only export the draft version.
3. After you have chosen the version you want, click [Export](#). The exported JSON file is automatically downloaded to your browser. You can find it in your local download file path.

### 3.6 Delete Model

#### Context

You can delete one model at a time.

#### Procedure

1. On the *Model Overview* page, select the model you want to delete, then click the [Delete](#) button.
2. A confirmation message box appears. You can choose to continue or abort.
3. If you continue, you get a success message when deletion is complete. If you meet an error, the [Last Operation Status](#) becomes [Deletion Error](#), and you can [view the model history](#) on the *Model Details* page to check the details.

## 4 CREATE AND DEPLOY A MODEL

### 4.1 Create a Model from Scratch

#### Context

MM app allows you to create a model from scratch.

#### Procedure

1. Launch the MM app.
2. On the *Model Overview* page, click the [Create](#) button.  
The detail page of a blank model then appears.
3. On the *Model Details* page, you can do the following:
  - fill in the [model header](#) information
  - create one or more [tracked processes](#), and in each tracked process create user-defined fields, planned events and unplanned events
  - create [field types](#): can be used in tracked process user-defined fields definition as composition type
  - create [event types](#): can be used in tracked process planned/unplanned event definition
  - create [code lists](#): can be used in tracked process user-defined fields definition as code list type
  - define the [IDOC integration](#) mapping rules for a tracked process
  - define the [Visibility Provider integration](#) mapping rules for a tracked process
  - create the [planned event extension fields](#)
  - define the '[Event to Action](#)' rules for the model.  
(This replaces the business rules used in core engine Version 1.)
4. On the *Model Details* page, click the [Save](#) button.

## 4.2 Edit an Existing Model

### Context

MM app allows you to import and edit a model.

### Procedure

1. Launch the MM app
2. On the *Model Overview* page, you can either:
  - import the model you want to edit by clicking the [Import](#) button
  - or select an existing model.The detail page of a model then appears.
3. On the *Model Details* page, you can do the following:
  - edit the [model header](#) information
  - edit existing [tracked processes](#) or create new ones, and in each tracked process edit / create user-defined fields, planned events and unplanned events
  - edit existing [field types](#) or create new ones: can be used in tracked process user-defined fields definition as composition type
  - edit/create [event types](#): can be used in tracked process planned/unplanned event definition
  - edit/create [code lists](#): can be used in tracked process user-defined fields definition as code list type
  - edit the [IDOC integration](#) mapping rules for a tracked process or define new ones
  - edit the existing [Visibility Provider integration](#) mapping rules for a tracked process or define new ones
  - edit existing [planned event extension fields](#) or create new ones
  - edit the existing '[Event to Action](#)' rules for the model or define new ones.
4. On the *Model Details* page, click the [Save](#) button.

## 4.3 Deploy a Model

### Context

When you have created a model, you can deploy the model directly.

### Procedure

1. Launch the MM app.
2. On the *Model Overview* page, click the model you want to deploy.  
The detail page of the model then appears.
3. On the *Model Overview* page, click the [Deploy](#) button on the top right. The model is then deployed.

#### 4.4 Model Header

##### Context

In the [Header](#) tab, you can fill in the following basic information about the model:

- Name
- Description
- Namespace
- Correlation Level

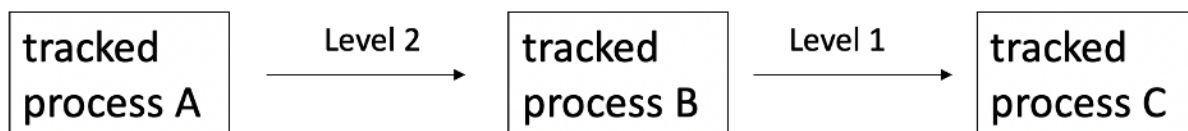
##### Correlation level:

- is the level of event correlation propagation between tracked processes within a model.
- is specified as an integer from 1 to 5.

Each tracked process has a list of observed tracked processes. You can add a tracked process to the observed tracked process list by the process creation/update event or by some other actual events.

For example, suppose there is an event correlation chain between tracked processes A, B, and C, which means that tracked process A observes tracked process B and tracked process B observes tracked process C.

- If the model is configured to have 1 level of correlation, then events reported to tracked process C are also correlated to B.
- If the model is configured to have 2 levels of correlation, then events reported to tracked process C are also correlated to both A and B.
- And so on.



Note that the event correlation chain between tracked processes breaks up in the following two cases:

- When tracked processes are of the same type.

In this example, if tracked processes A and C are of the same type, then the event correlation chain breaks up between B and A. So events reported to tracked process C are correlated to B, but not to A.

- When the actual event's event type is not defined as admissible planned or unplanned event in this tracked process.

In this example, if a Proof of Delivery planned/unplanned event has been defined in tracked processes C and A, but not in B, then when the Proof of Delivery actual event is reported to C, the event correlation chain breaks up between B and C. So the Proof of Delivery actual event reported to C won't be correlated to B and A, even if there is a Proof of Delivery planned/unplanned event defined in A.

Also note that the authorization for instances can also be inherited and checked through correlation. So in this example,

- If you have authorization for tracked process C, you will also have authorization for B and A. Then events reported to C are also correlated to B and A.
- If you don't have authorization for tracked process C, the system will check if you have authorization for B. If so, events will be reported to B and correlated to A. Otherwise, the system will check if you have authorization for A.
- If tracked process C doesn't exist, the system will check if you have authorization for B. If so, events will be reported to B and correlated to A. Otherwise, the system will check if you have authorization for A.

## Guide for Model Administrators

To complete the model header, do the following.

### Procedure

1. Fill in the mandatory field: [Name](#). It must:
  - be unique from other models in the same tenant
  - start with a lowercase letter and then lowercase letters or numbers
  - not start with letters “gtt” and
  - be less than 50 characters.
2. When the model name is set, the [Namespace](#) will be filled automatically.
3. Fill in the [Description](#).
4. Fill in the [Correlation Level](#) in the range from 1 to 5.

## 4.5 Tracked Process

### Context

A tracked process is the main object in the model. Each model must have at least one tracked process. The name of the tracked process must be unique in the model. You can create/edit/delete your own tracked process. You can have at most 200 tracked processes in a model.

A tracked process includes:

- a name (mandatory)
- description
- tracking ID type (mandatory)
- user-defined field list
- admissible planned events and
- admissible unplanned events.

In the [Tracked Process](#) tab, user-defined tracked processes are listed on the left. Select one tracked process in the list. On the right, there are different views of the tracked process:

- [Field Lists](#)
- [Admissible Planned Events](#) and
- [Admissible Unplanned Events](#).

### 4.5.1 Create Tracked Process

### Context

You can create a tracked process in the [Tracked Process](#) tab.

### Procedure

1. In the [Tracked Process](#) tab, click the [Create](#) button above the list.
2. Fill in the following:
  - [Name](#): mandatory and must be unique in this model.
  - [Description](#)
  - [Tracking ID Type](#): mandatory and must be unique across models in the whole tenant
  - [Affect Planned Event Status By](#): mandatory. You must select in the drop-down list the timestamp, either Actual Business or Actual Technical. The default is [Actual Business Timestamp](#). Your selection decides the timestamp the system uses to process the last reported actual event. In turn, this affects the planned event status. For more information, see [Timestamps](#).
  - [Sort Planned Events By](#): mandatory and you can select either [First Planned Business Timestamp then Payload Sequence](#) or [First Payload Sequence then Planned Business Timestamp](#) in the drop-down list. The first one is the default. Your selection decides the sequence of planned events used for process status calculation. You can see the displayed sequence in the [Planned Events](#) tab of the [CPE](#) app.
3. Click [OK](#).

#### **4.5.2 Edit Tracked Process**

##### **Context**

After a tracked process is created, you can edit the tracked process attributes such as name, description and tracking ID type.

##### **Procedure**

1. In the [Tracked Process](#) tab, click the tracked process that you want to change.
2. Click the [Edit](#) button above the list.
3. In the popup dialog, you can edit its name, description and tracking ID type, and change your selection in [Affect Planned Event Status By](#) and [Sort Planned Events By](#).
4. When editing is complete, click [OK](#) to save your changes in the model. Model detail page is changed to the edit mode.
5. Click [Save](#) to save the changes.

#### **4.5.3 Delete Tracked Process**

##### **Context**

You can delete a Tracked Process.

##### **Procedure**

1. In the [Tracked Process](#) tab, click the event in the list.
2. Click the [Delete](#) button above the list.
3. There are two situations:
  - If this tracked process is not used by other objects, before deletion, a warning message box pops up to ask you if you want to delete the tracked process and its fields such as user defined field list, admissible planned event, and admissible unplanned event.
  - Otherwise, a [Where Used List](#) pops up to ask you if you want to delete the tracked process and all the data in the places where this tracked process is used.
4. If you click the [OK](#) button, deletion takes effect immediately and the model detail page is changed to the edit mode.



#### 4.5.4 Field List

##### Context

In the [Tracked Process](#) tab, click content, Field List, two tables appear.

- The first table contains the user fields in this tracked process. You can create/edit/delete these user fields for this tracked process.
- The second table contains all the fields from the pre-defined core model process types. You cannot modify them.

You can have at most 200 fields in a tracked process.

##### Procedure

1. In the [Tracked Process](#) tab, click content, Field List, create one user field
  - a. Click the [Create](#) button above the user fields table
  - b. Fill in the following information:
    - i. Name: unique in one tracked process, includes fields in core model process types
    - ii. Label
    - iii. Type:
      - ◇ String: maximum length is 4000
      - ◇ UUID
      - ◇ Boolean
      - ◇ Integer
      - ◇ Decimal
      - ◇ Date
      - ◇ Timestamp
      - ◇ Association to One: associates to a tracked process, linked to a tracked process
      - ◇ Association to Many: associates to more than one tracked process, based on Association to One fields. If you choose this type, you must select Process Type and Field (Association to One). For the Field (Association to One), you can also input values freely to create new fields.
      - ◇ Composition: item level table with structure defined in field type
      - ◇ Code List: identifies a code list with string type and value range
      - ◇ Attachments:
        - a. Supported attachment file types are: jpeg, jpg, gif, png, pdf, doc, docx, xls, and.xlsx.
        - b. The maximum size limit is 10MB.
        - c. For IDOC Integration, attachments fields don't support IDOC mapping. For Visibility Provider Integration, only the event's attachments fields support VP mapping.
    - iv. DPP: Abbreviation for data protection and privacy. It includes the following selectable values: Data Subject ID and Personal Identifiable Information.
      - ◇ There must be only one Data Subject ID in one tracked process.
      - ◇ Fields annotated as Data Subject ID include the properties of Personal Identifiable Information.
      - ◇ Changes to the fields annotated as Data Subject ID trigger updates of the PDM data visibility, Audit Log Viewer read log and Audit Log Viewer change log.
      - ◇ Changes to fields annotated as Personal Identifiable Information trigger updates of the PDM data visibility and Audit Log Viewer change log.
      - ◇ For more information, refer to the chapters "Security: General" and "Security: Data Protection and Privacy" in the *Administration Guide for Solution Owners*.
    - v. Grant: Only the String type can be granted Read or Report.
      - ◇ Read: If this field select Read and user LBN ID is equal to this field, then you can read this tracked process where the field exists.
      - ◇ Report: If this field select Report and user LBN ID is equal to this field, then you can write this tracked process where the field exists.

- vi. Role Attribute: is specified to a string or code list field to check if a user role has authorization to read this tracked process based on the value of the role attribute defined in SAP BTP cockpit.
    - ◇ You can assign at most nine attributes to nine different fields in a tracked process.
    - ◇ Select in the drop-down list from gttUserAuthAttribute001 to gttUserAuthAttribute009.
    - ◇ To use this function, you also need to add the role "ReadServiceAttributeTemplate" to the role collection and specify attributes in this role in SAP BTP cockpit. For more, see the section "[Specify Attributes in the Role Using "ReadServiceAttributeTemplate"](#)" in *Administration Guide for Solution Owners*.
  - vii. Readable: when reading the tracked process, this field is readable by Odata.
  - viii. Writable: when writing the tracked process, this field is writable.
  - c. Click the **OK** button to create this field
2. Edit one user field: supports single field edit at one time.
- a. Click the pencil icon at the end of the field that needs to be changed in the table
  - b. Edit in the popup dialog, then click the **OK** button to save changes
3. Delete user fields: supports multi-field deletion in this model.
- a. Select the fields you want to delete.
  - b. Click the **Delete** button.
  - c. A **To Be Deleted Objects** dialog pops up to ask you if you want to delete the selected fields and all the data in the places where the fields are used.
  - d. Click the **OK** button. Fields you selected are removed immediately and the model detail page is changed to the edit mode.
4. Add a URL for one user field:
- a. In the edit mode, select a field whose type is not Composition, Attachments, and Association to Many.
  - b. Click the **Add URL** button.
  - c. Fill in the name and address. The address supports variables. Variables should be in curly brackets, such as {Field1}. The following fields can be used as variables:
    - i. A user field with the type that is not Composition, Attachments, and Association to Many in this tracked process.
    - ii. A user field with the type that is not Composition, Attachments, and Association to Many in the associated tracked process, if the type of the field that the URL is added for is Associate to One.

Note:

If you want to navigate to the external systems from the fulfillment tracking apps, you must add URLs for user fields in the standard model "gttft1".

You can add either of the following two types of address:

- address with a fixed URL prefix: for example, [www.sap.com/?zzdeliveryNo={variable}](http://www.sap.com/?zzdeliveryNo={variable}).
- address with a URL prefix variable: for example, [{URL prefix variable}/?zzdeliveryNo={variable}](http://{URL prefix variable}/?zzdeliveryNo={variable}).

5. Delete a URL for one user field:
- a. In the edit mode, select a field that you have added a URL.
  - b. Click the **Add URL** button.
  - c. Select the URL that you want to delete.
  - d. Click the Delete icon. The URL is removed immediately.
  - e. Click **OK**.
  - f. Click **Save** to save changes.

#### 4.5.5 Admissible Planned Events

##### Context

To define planned events, you can use admissible planned events. You can create/edit/delete admissible planned events in the [Tracked Process](#) tab by selecting [Admissible Planned Events](#) on the top right of this view and add planned event properties. All the admissible planned events are listed on the right side of this page. You can have at most 200 admissible planned events in a tracked process.

##### Create Admissible Planned Events

##### Procedure

1. Click the [Add](#) button above the planned event list.
2. A popup dialog appears. There are three tabs: [Details](#), [Match Actual Event](#), and [Match Planned Event](#).
3. In the [Details](#) tab:
  - a. Select an event type that you defined in the [Event Type Pool](#) tab. If the event type is already used by another admissible planned event in the same tracked process, this event type is not selectable and you are not able to select it as an admissible planned event again. Note that the events that inherit GTTDelayedEvent and GTTOnTimeEvent won't be displayed in the drop-down list.
  - b. Fill in the following:
    - *Default Business Tolerance*: used to calculate the earliest planned business timestamp and the latest planned business timestamp, if they are not provided in the event payload. The default value is 0. Enter a number that is greater than or equal to 0.
    - *Default Technical Tolerance*: used to calculate the earliest planned technical timestamp and the latest planned technical timestamp, if they are not provided in the event payload. The default value is 0. Enter a number that is greater than or equal to 0.
    - *Periodic Overdue Detection*: used to define how often you want the system to check the status of overdue events. If you enter 10 minutes, then the system will check the status of overdue events every 10 minutes. Enter a number that is greater than or equal to 5.
    - *Max Overdue Detection*: the maximum times that you want the system to check the status of overdue events. If you enter 3, the system will check the overdue events three times and then ignore them even if they are still overdue. Enter a number that is less than or equal to 5.
    - *Affect Planned Event Status By*: Select in the drop-down: actual business timestamp, actual technical timestamp, or inheriting the configuration from your tracked process definition. The default is *Actual Business Timestamp*. Your selection decides the timestamp the system uses to process the last reported actual event. In turn, this affects the planned event status. For more information, see [Timestamps](#) and [Create Tracked Process](#).
4. In the [Match Actual Event](#) tab, you can define how an actual event matches an admissible planned event when an actual event is sent to the GTT system:
  - a. Define the match location value by clicking the switch control.
    - On: Location is used as a match key between the planned event and actual event. The check box of [locationAltKey](#) is enabled in the [Match Planned Event](#) tab.
    - Off: Location is not a match key between the planned event and actual event. The check box of [locationAltKey](#) is disabled in the [Match Planned Event](#) tab.

- b. Below the location match condition, you can define match conditions between planned event extension fields and actual event fields. You can add/edit/delete a match condition. The field you added/deleted here will also enable/disable the check box of this field displayed in the [Match Planned Event](#) tab.

Note: If a planned event extension field you used here doesn't exist in the event types that inherit GTTDelayedEvent, then the system will add this extension field to the event types that inherit GTTDelayedEvent.

- Click the Add icon to add a match condition.
  - On the left, choose a planned event extension field you defined in the [Planned Event Extension](#) tab, the match operator is set as equal by default. On the right, choose an actual event field. The two match fields must be of the same type.
  - Click the [Delete](#) icon on the right to delete a corresponding match condition.
5. In the [Match Planned Event](#) tab, you can define how a planned event matches an existing admissible planned event when a planned event update or tracked process update is sent to the GTT system:
    - a. You can define the match value by using the check box. The check box of [eventType](#) is enabled by default and cannot be changed. Fields that are not defined in the [Match Actual Event](#) tab won't be displayed here. Note that if the match value is not maintained here, the GTT system will fall back to the match conditions in the [Match Actual Event](#) tab when matching planned events.
  6. Click [OK](#) to close the dialog and add a planned event if all checks in the dialog pass.

Note:

In the GTT standard model "gttt1", you cannot change event type, event match key, and standard fields in the [Match Actual Event](#) and [Match Planned Event](#) tabs for the events below:

- standard admissible planned events
- user-defined admissible planned events with a standard event type.

### ***Edit Admissible Planned Events***

#### **Procedure**

1. Click the pencil icon at the end of the field that needs to be changed in the table.
2. Edit the admissible planned event in the dialog.
3. Click [OK](#) to save changes.

### ***Delete Admissible Planned Events***

#### **Procedure**

1. Select the admissible planned event that you want to delete. Support multi deletion.
2. Click the [Delete](#) button.
3. Admissible planned event(s) you selected is/are removed.

#### 4.5.6 Admissible Unplanned Events

##### Context

You can use admissible unplanned events to define unplanned events. You can create/edit/delete admissible unplanned events in the [Tracked Process](#) tab by selecting *Admissible Unplanned Events* on the top right of this view. You can have at most 200 admissible unplanned events in a tracked process.

On this page, all the:

- admissible unplanned events are listed on the top right and
- core unplanned event types are listed on the bottom right.

##### Create Admissible Unplanned Events

##### Procedure

1. Click the [Add](#) button above the unplanned event list.
2. A popup dialog appears. There are two tabs: [Details](#) and [Match Actual Event](#).
3. In the [Details](#) tab, select the event type you defined in the [Event Type Pool](#) tab.
4. In the [Match Actual Event](#) tab, you can define how an actual event matches an admissible unplanned event when an actual event is sent to the GTT system. Here, you can add/edit/delete a match condition between
  - a. an actual event field and a string field. Only constants, empty and null fields are allowed.
  - b. an actual event field and a tracked process field. The two match fields should be of the same type. Attachments, composition, and association types are excluded. If the type is code list, the two match fields must come from the same code list. Tracked process fields only support standard and user-defined fields.
  - c. an actual event field and a subfield in the tracked process composition field. The match condition works when the actual event field matches one of the subfields of the composition field.
5. Click the [OK](#) button to close dialog and add an unplanned event if all checks in the dialog pass.

Note:

The unplanned event match settings don't apply to unplanned events of which the event type is GTTUpdatePlanEvent.

If the referred planned event in the actual event payload is found, the settings also don't apply to the following actual events:

- actual events of which the event type is GTTOnTimeEvent or GTTDelayedEvent;
- actual events that inherit GTTOnTimeEvent or GTTDelayedEvent.

##### Edit Admissible Unplanned Events

##### Procedure

1. Click the pencil icon at the end of the field that needs to be changed in the table.
2. Edit the admissible planned event in the dialog.
3. Click the [OK](#) button to save changes.

##### Delete Admissible Unplanned Events

##### Procedure

1. Select the admissible planned event that you want to delete. Support multi deletion
2. Click the [Delete](#) button.
3. Admissible planned event(s) you selected is/are removed.

## 4.6 Field Type

### Context

You can create/edit/delete user-defined [Field Type](#) in the [Field Type Pool](#) tab. It is used to define the composition of user-defined fields for [Tracked Process](#), [Field Type](#) and [Event Type](#). You can have at most 200 field types in a model.

In the [Field Type Pool](#) tab, the Field Types are listed on the left. The User Fields table on the right changes according to the Field Type you select on the left.

### 4.6.1 Create Field Type

### Procedure

1. Click the [Create](#) button above the Field Type List.
2. Select the target [Tracked Process](#).
3. Fill in the [Name](#) of the Field Type. The name must be unique in the model.
4. Click the [OK](#) button to close the dialog if all the checks pass.
5. On the right side, a User Model Fields table appears. It contains user fields that only belong to this field type. You can have at most 200 user fields in a field type. You can create/edit/delete user fields for this event type.
  - a. Create one user field
    - i. Click the [Create](#) button above user fields table.
    - ii. Fill in name, label, select type, and define this field's scope by using the check box. The name must be unique in one field type.
    - iii. Click the [OK](#) button to create this field.
  - b. Edit one user field
    - i. Click the pencil icon at the end of the field that needs to be changed in the table.
    - ii. Edit in the popup dialog, then click the [OK](#) button to save changes.
  - c. Delete user fields: supports multi-field deletion.
    - i. Select the fields you want to delete.
    - ii. Click the [Delete](#) button.
    - iii. A [To Be Deleted Objects](#) dialog pops up to ask you if you want to delete the selected fields and all the data in the places where the fields are used.
    - iv. Click the [OK](#) button. Fields you selected are removed immediately and the model detail page is changed to the edit mode.
  - d. Add a URL for one user field:
    - i. In the edit mode, select a field whose type is not Composition, Attachments, and Association to Many.
    - ii. Click the [Add URL](#) button.
    - iii. Fill in the name and address. The address supports variables. Variables should be in curly brackets, such as {Field1}. The following fields can be used as variables:
      - A user field with the type that is not Composition, Attachments, and Association to Many in this tracked process.
      - A user field with the type that is not Composition, Attachments, and Association to Many in the associated tracked process, if the type of the field that the URL is added for is Associate to One.

#### Note:

If you want to navigate to the external systems from the fulfillment tracking apps, you must add URLs for user fields in the standard model "gtttf1".

You can add either of the following two types of address:

- address with a fixed URL prefix: for example, [www.sap.com/?zzdeliveryNo={variable}](#).
- address with a URL prefix variable: for example, [{URL prefix variable}/?zzdeliveryNo={variable}](#).

- e. Delete a URL for one user field:
  - i. In the edit mode, select a field that you have added a URL.
  - ii. Click the [Add URL](#) button.
  - iii. Select the URL that you want to delete.
  - iv. Click the Delete icon. The URL is removed immediately.
  - v. Click [OK](#).
  - vi. Click [Save](#) to save changes.

#### **4.6.2 Edit Field Type**

##### **Context**

You can change the Field Type Name and the Tracked Process it belongs to.

##### **Procedure**

1. Click the [Edit](#) button above the Field Type List.
2. Change the selection of the target [Tracked Process](#).
3. Change the [Name](#) of the Field Type.
4. Click the [OK](#) button to close the dialog if all the checks pass.

#### **4.6.3 Delete Field Type**

##### **Context**

You can delete a Field Type.

##### **Procedure**

1. Select the target Field Type.
2. Click the [Delete](#) button above the Field Type List.
3. The target Field Type is removed if it is not referred by other objects.
4. Otherwise, before deletion, a [Where Used List](#) pops up to ask you if you want to delete the field type and all the data in the places where this field type is used.
5. If you click the [OK](#) button, deletion takes effect immediately and the model detail page is changed to the edit mode.

## 4.7 Event Type

Event type is used in the planned events and unplanned events of a tracked process. There are two types:

- core model event type and
- user-defined event type.

You can create/edit/delete your own user-defined event type, including the event type name, description, inherit event type, field list of the event type. You can have at most 200 event types in a model.

In the [Event Type Pool](#) tab, user-defined event types and core model event types are listed on the left. Select an event type in the list. Then on the right side, the field list is displayed in two tables. The table at the:

- top contains all user-defined fields used in this event type
- bottom contains fields inherited from core model event type

### 4.7.1 Create Event Type

#### Context

You can create an event type in a tracked process, including attributes such as event type name, description, inherit event and field list.

#### Procedure

1. Click the [Create](#) button above the list on the left.
2. Select a tracked process in the drop-down list.
3. Fill in the event type name and description. The name is mandatory and should be unique in the model. In the Inherit drop-down list, default value Event is provided. You can also change it based on your requirement.
4. Click the [OK](#) button to close dialog if all checks pass.
5. On the right side, two tables appear. The first contains user fields that only belong to this event type. The other contains all fields from the event this event type inherits. You can have at most 200 user fields in an event type.  
You can create/edit/delete user fields for this event type.
  - a. Create one user field
    - i. Click the [Create](#) button above user fields table.
    - ii. Fill in name, label, select type, and define this field's scope by using the check box. The name must be unique in one event type and include fields in core model event type inherit. For event types that inherit GTTDelayedEvent, if a field has the same name with a planned event extension field, its properties such as type will be copied from the planned event extension field. The properties cannot be changed.
    - iii. Click the [OK](#) button to create this field.
  - b. Edit one user field:
    - i. Click the pencil icon at the end of the field that needs to be changed in the table.
    - ii. Edit in the popup dialog, then click the [OK](#) button to save changes.
  - c. Delete user fields: supports multi-field deletion.
    - i. Select the fields you want to delete.
    - ii. Click the [Delete](#) button.
    - iii. A [To Be Deleted Objects](#) dialog pops up to ask you if you want to delete the selected fields and all the data in the places where the fields are used.
    - iv. Click the [OK button](#). Fields you selected are removed immediately and the model detail page is changed to the edit mode.
  - d. Add a URL for one user field:
    - i. In the edit mode, select a field whose type is not Composition, Attachments, and Association to Many.
    - ii. Click the [Add URL](#) button.
    - iii. Fill in the name and address. The address supports variables. Variables should be in curly brackets, such as {Field1}. The following fields can be used as variables:
      - A user field with the type that is not Composition, Attachments, and Association to Many in this tracked process.



- A user field with the type that is not Composition, Attachments, and Association to Many in the associated tracked process, if the type of the field that the URL is added for is Associate to One.

Note:

If you want to navigate to the external systems from the fulfillment tracking apps, you must add URLs for user fields in the standard model “gttt1”.

You can add either of the following two types of address:

- address with a fixed URL prefix: for example, [www.sap.com/?zzdeliveryNo={variable}](http://www.sap.com/?zzdeliveryNo={variable}).
- address with a URL prefix variable: for example, [{URL prefix variable}/?zzdeliveryNo={variable}](http://{URL prefix variable}/?zzdeliveryNo={variable}).

- e. Delete a URL for one user field:
  - i. In the edit mode, select a field that you have added a URL.
  - ii. Click the [Add URL](#) button.
  - iii. Select the URL that you want to delete.
  - iv. Click the Delete icon. The URL is removed immediately.
  - v. Click [OK](#).
  - vi. Click [Save](#) to save changes.

### 4.7.2 Edit Event Type

#### Context

After the event type has been created, you can edit its attributes including name, description, inherit event, and field list.

#### Procedure

1. Click the event type you want to change on the left side.
2. Click the [Edit](#) button above list.
3. In the popup dialog, you can edit its Tracked Process, Name, Description and Inherit fields.
4. After editing is complete, click the [OK](#) button and save changes to the model. And the *Model Details* page is changed to edit mode.

### 4.7.3 Delete Event Type

#### Context

You can delete an Event Type.

#### Procedure

1. Select the target Event Type.
2. Click the [Delete](#) button above the Event Type list.
3. The target Event Type is removed if it is not used in other places.
4. Otherwise, before deletion, a [Where Used List](#) pops up to ask you if you want to delete the event type and all the data in the places where this event type is used.
5. If you click the [OK](#) button, deletion takes effect immediately and the model detail page is changed to the edit mode.

## 4.8 Code List

You can create/edit/delete code list in a tracked process. Each code list is a collection of code with a name. The code list is consumed as field type in field definition of tracked process, field type, event type and planned event. Code list fields can receive empty value. You can have at most 200 code lists in a model.

In the [Code List](#) tab, code lists are displayed in alphabetical order. Click a code list, then the table on the right displays the corresponding collection of code and name.

### 4.8.1 Create Code List

#### Context

Code list is used in field definition as field type. Name of code list should be unique in one model. Code list includes its name and codes.

#### Procedure

1. Click the [Create](#) button above the list
2. Fill into the name
3. Click the [OK](#) button to close dialog
4. After creation is complete, you can create/edit/delete code for this code list. You can have at most 200 codes in a code list.
  - a. Create Code in Code List. Code should be unique in one code list.
    - i. Click the [Add](#) button above the table.
    - ii. A new row with two empty inputs will be added on the bottom of the table.
    - iii. Fill in code and name. If code is left empty, this line will not be saved to model.
    - iv. Click the [Save](#) button to save changes to model.
  - b. Edit Code in Code List
    - i. Click [Edit](#) on the top right of MM app to change model detail view to edit mode.
    - ii. Change name and code in the table.
    - iii. Click [Save](#) to save changes to model.
  - c. Delete Code in Code List
    - i. Click [Edit](#) on the top right of MM app to change model detail view to edit mode.
    - ii. In the codes table, click the [Delete](#) icon at the end of the line you want to delete.
    - iii. The deletion takes effect immediately.
    - iv. Click [Save](#) to save changes to model.
  - d. Import Code in Code List
    - i. Click the [Import](#) button above the table.
    - ii. A popup appears. Select an [LBN Code List](#) that you want to import. You can preview the codes of the selected LBN code list in the table below.
    - iii. Click [OK](#) to import the codes.
    - iv. Click [Save](#) to save changes to model.

#### Note

The following LBN code lists are provided:

- Carrier Reference Document Type
- Event Reason Code Descriptions
- Incoterm Descriptions
- Location Type Descriptions
- Shipper Reference Document Type
- Shipping Type Descriptions
- Traffic Direction Descriptions
- Transportation Means Standard Code Descriptions
- Transportation Modes

For details, search for the document [Code Lists](#) at [help.sap.com/qtt](https://help.sap.com/qtt).

#### **4.8.2 Edit Code List**

##### **Context**

You can edit a code list's name.

##### **Procedure**

1. Click the code list item you want to change.
2. Click the [Edit](#) button above the list.
3. In the popup dialog, you can edit its Name.
4. After changes are complete, click the [OK](#) button to save changes to model. And the *Model Details* page is changed to edit mode.

#### **4.8.3 Delete Code List**

##### **Context**

You can delete a Code List.

##### **Procedure**

1. Select the target Code List.
2. Click the [Delete](#) button above the list.
3. If the code list is not used in other places, deletion takes effect immediately and the model detail page is changed to the edit mode.
4. Otherwise, before deletion, a [Where Used List](#) pops up to ask you if you want to delete the code list and all the data in the places where this code list is used. If you click the [OK](#) button, deletion takes effect immediately and the model detail page is changed to the edit mode.

#### 4.9 IDOC Integration

IDOC Integration maintains mapping rules between ERP IDOC and tracked process user-defined fields. It defines:

- ERP Object Type
- Application Object Type
- the mapping of tracked process event and IDOC
- the mapping of tracked process actual event and ERP Event Code
- the mapping of events' IDOC Segment/IDOC Field and tracked process user-defined fields

On the [IDOC Integration](#) tab, you can select a specific tracked process to display or edit its related properties.

On the right of the tracked process selection, there is an IDOC integration switch for this tracked process. The default state of this switch is OFF. You can change the state in edit mode. When the switch state is:

- OFF, the related configuration view is not visible. These mapping properties are not checked.
- ON, you can see below the tracked process mapping, an event type list and a fields list tree table.

When saving a draft model, these selected and filled items are checked before saving or deploying it.

In the tracked process mapping part, you can select *ERP Object Type* and fill in *Application Object Type*.

##### ERP Object Type

You can select:

- Purchase Order Header
- Purchase Order Item
- Inbound Delivery Header
- Inbound Delivery Item
- Sales Order Header
- Sales Order Item
- Outbound Delivery Header
- Outbound Delivery Item
- Freight Unit
- Shipment
- Others

Your selection can decide the F4 Help lists of Event Code, IDOC Segment, and IDOC Field.

##### Application Object Type

This is a mandatory field. Your entry specifies the application object type in the ERP system for IDOC Integration.

##### Tracked Process / Events List

This includes tracked process event and event types. You are free to input your own value or use value help to define IDOC and ERP Event Code. Event code must be unique in a model.

##### User Model Fields List

There are Field column, IDOC segment column and IDOC field column in the tree table. You are free to input your own values or use value help to define related properties according to your selected ERP Object Type.

#### **4.9.1 To Setup IDOC Integration**

##### **Procedure**

1. In the [IDOC Integration](#) tab, select a tracked process.
2. Set the *Integration Switch* to ON for IDOC Integration.
3. Select an *ERP Object Type*.
4. Fill in the *Application Object Type*.
5. *IDOC* in the event type list: if the *ERP Object Type* value is:
  - “Others”, use value help or input a value manually
  - otherwise, the corresponding is pre-entered for you
6. *ERP Event Code* in the event type list: use value help or input a value manually.
7. *IDOC Field* and *IDOC Segment*: use value help or manually enter your own values.

#### 4.10 Visibility Provider Integration

Visibility Provider Integration is used for mapping visibility provider generic interface with a tracked process and its actual events. On the [Visibility Provider Integration](#) tab, you can select a specific tracked process to display or edit related properties.

On the right of the tracked process selection is an [Integration Switch](#) for this tracked process. The default state of this switch is OFF. You can change the state in Edit mode. When the switch state is:

- OFF, the related configuration view is not visible. These mapping properties are not checked.
- ON, you can see below the tracking indicator, the [Generate Event for Tracking Request Failure](#) switch, an event type list, and a fields list tree table.

When saving a draft model, these selected and filled items are checked before saving or deploying it.

##### Tracking Indicator

Tracking indicator is sent to the visibility provider. It has two fixed values:

- 01(Shipment tracking): to show that the tracked process is mapped to the shipment tracking related service in the Configure Partner Connections app.
- 02(Resource tracking)

If you select 02(Resource tracking), Source Tracked Process is visible and able to input the tracked process type whose tracking indicator is 01(Shipment tracking). Otherwise, it is not visible and disabled.

##### Generate Event for Tracking Request Failure

This switch allows you to automatically generate an event called “[GTT Tracking Request Error Event](#)” when your tracking request is rejected by your business partner. The event contains an error message that your business partner sent to you. You can use the information for troubleshooting.

The event will be displayed in:

- the Check Processes and Events app’s [Other Events](#) table, which contains a core model field [Error Message](#) to display the error.
- the fulfillment tracking apps’ [Events](#) table, which contains a field [Event Reason Text](#) to display the error. You can also view this error in the [Error Message](#) field of the [Event Details](#) page. For more, see *Guide for Transportation Planners*.

##### Source Tracked Process

You can select one or more tracked Process with 01(Shipment tracking) as ‘Source Tracked Process’. When this tracked process is forwarded to the visibility provider by its Write Service, it triggers the Write tracked process.

##### Tracked Process / Events List

In the Tracked Process / Events list, there are tracked process events and event types. For event types, you can use value help to define “LBN Event Type”.

To see the full list of supported LBN event types, search for the document *Code Lists* at [help.sap.com/gtt](https://help.sap.com/gtt) and see the section “[Event Code Descriptions](#)”.

##### Standard/User Model Fields List

- Tracked Process event:  
The field list includes LBN Tracking fields list and tracked process fields column. You can select the tracked process field. You cannot input values freely in edit mode.
- Event Type event:  
The field list includes event user define field column and LBN Tracking fields column. You can select the LBN Tracking field. You cannot input values freely in edit mode.

Note:

In the standard model “gttt1”, you can:

- change the mapping of standard event types with LBN event types. But if you do so, you will no longer benefit from future upgrades of the standard model.
- map user-defined event types with LBN event types. But if you do so, you will no longer benefit from future upgrades of the standard model.

- change the mapping of the tracked process's standard fields with the LBN tracking fields. But if you do so, you will no longer benefit from future upgrades of the standard field mapping of this tracked process.

If you change the standard field mapping of the “Shipment” and “TrackingUnit” tracked processes, you can refer to the tables below for the upgrades, and manually update the field mapping accordingly.

***Shipment Tracked Process:***

<b>LBN Tracking Fields</b>	<b>Field</b>
Carrier LBN ID	serviceAgentLbnId
Ordering Party Name	
Consignee Name	
Quantity of Total Pieces	
Quantity of Total Pieces Unit	
Transportation Mode	transportationMode
Transportation Means Standard Code	transportMeans
Transportation Means Tracked Objects	trackedObjects
• ID	idType
• Value	value
Dangerous Goods Indicator	dangerousGoods
Incoterm	incoterms
Incoterm Location	incotermLocation
Cargo Gross Weight	
Cargo Gross Weight Unit	
Cargo Gross Volume	
Cargo Gross Volume Unit	
Carrier Reference Document	carrierRefDocuments
• Document Type	docType
• Document	docId
Shipping Type	shippingType
Document Reference	shipperRefDocuments
• Document Type	docType
• Document	docId
Total Number of Stops	
Total Number of Loading Locations	
Total Number of UnLoading Locations	
Stops	stops
• Stop ID	stopId
• Ordinal No	ordinalNo
• Location Object Type	locationType
• Location ID	locationId
• Location Alternative Key	locationAltKey
Stages	
• Stage ID	
• Loading Stop ID	
• Unloading Stop ID	



**TrackingUnit Tracked Process:**

LBN Tracking Fields	Field
Carrier LBN ID	serviceAgentLbnId
Ordering Party Name	
Consignee Name	
Quantity of Total Pieces	
Quantity of Total Pieces Unit	
Transportation Mode	transportationMode
Transportation Means Standard Code	
Transportation Means Tracked Objects	trackedObjects
• ID	idType
• Value	value
Dangerous Goods Indicator	
Incoterm	
Incoterm Location	
Cargo Gross Weight	
Cargo Gross Weight Unit	
Cargo Gross Volume	
Cargo Gross Volume Unit	
Carrier Reference Document	carrierRefDocuments
• Document Type	docType
• Document	docId
Shipping Type	shippingType
Document Reference	
• Document Type	
• Document	
Total Number of Stops	
Total Number of Loading Locations	
Total Number of UnLoading Locations	
Stops	stops
• Stop ID	stopId
• Ordinal No	ordinalNo
• Location Object Type	locationType
• Location ID	locationId
• Location Alternative Key	locationAltKey
Stages	
• Stage ID	
• Loading Stop ID	
• Unloading Stop ID	

**4.10.1 To Setup Visibility Provider Integration****Procedure**

1. Select a tracked process and switch on visibility provider integration.
2. Select tracking indicator.
3. If the tracking indicator is 02, select source tracked process.
4. Use value help to fill in the LBN event type for actual events.
5. Use value help to fill in the field or LBN tracking field.

## 4.11 Planned Event Extension

### Context

Planned event extension allows you to define your own planned event extension fields. These fields are applied to all the planned events in the model. You can create/edit/delete user defined planned event extension fields in the [Planned Event Extension](#) tab. You can have at most 200 planned event extension fields in a model.

#### 4.11.1 Create Planned Event Extension Fields

### Procedure

1. Click the [Create](#) button above the user fields table.
2. Fill in the following:
  - name (must be unique in the planned event extension)
  - label
  - type (supports String, UUID, Boolean, Integer, Decimal, Date, and Timestamp)
  - readable and
  - writable.
3. Click the [OK](#) button to create this field.

#### 4.11.2 Edit Planned Event Extension Fields

### Context

You can edit the name, label, type, and readable or writable attributes of a planned event extension field.

### Procedure

1. Select the field that needs to be changed in the table.
2. Click the [Edit](#) button above the table. Or click the pencil icon at the end of the field that needs to be changed in the table.
3. Edit in the popup dialog, then click the [OK](#) button to save changes. And the *Model Details* page is changed to edit mode.

#### 4.11.3 Delete Planned Event Extension Field

### Context

You can delete a user model field of Planned Event Extension. Multi-field deletion is supported.

### Procedure

1. Select the fields you want to delete.
2. Click the [Delete](#) button above the list.
3. A [To Be Deleted Objects](#) dialog pops up to ask you if you want to delete the selected fields and all the data in the places where the fields are used.
4. Click the [OK](#) button. Fields you selected are removed immediately and the model detail page is changed to the edit mode.

## 4.12 Event to Action (Business Rules)

Under the [Event to Action](#) tab, you can add a script incorporating logic (business rules) to your model. For more information about the script, see "[Appendix: Event to Action Script](#)".

In the standard model "gttf1", the script is divided into four sections: Standard Declaration, User Script Before Standard Script, Standard Script, and User Script After Standard Script. Standard sections are not editable. You can only edit the user script sections. In the display mode, you can click the eyeglass icon to view the script in a popup.

### 4.12.1 Add a Script to Your Model

#### Context

To add a script to your model, do the following.

#### Procedure

1. Open the [Event to Action](#) tab.
2. You can click [Help](#) to open the documentation and find information on writing scripts.
3. Click the Edit button on the top right. Write or paste your script in the window. You can click the pencil icon to edit the script in a popup.
4. Click [Validate Code](#) to check your script for errors.
  - If there are no errors, you get a success message.
  - If an error is found, click [Open Validation Details](#) to see the line number and type of error.
  - Once you have corrected the error, click [Validate Code](#) again.
5. When you have finished your script, you have several options:
  - [Cancel](#): you can choose whether to *Discard all changes* and return to the previous script, or not.
  - [Save](#): to save your script in the model.

### 4.13 Translation

In the MM app, the following model metadata fields support multi-language translation:

- Description of Model
- Label of user field in Planned Event Extension, Tracked Process, Field Type and Event Type
- Description of tracked process
- Description of event type
- Name of code in code list

#### 4.13.1 Supported Language

English (US) is the default language and the default fallback language.

To see the full list of supported languages, see the section “Product Availability” in *Administration Guide for Solution Owners*.

#### 4.13.2 Translation Dialog

The translation dialog allows you to manage the translation of different languages.

##### Procedure

1. In the *Model Details* page, click the [Edit](#) button.
2. Click the [Tracked Process](#) tab.
3. To maintain a translation for a tracked process description:
  - a. Select a tracked process from the list.
  - b. Click the [Edit](#) button.
  - c. In the pop-up dialog,
    - i. If a translation of the description exists in the current logon language, it is displayed in the [Description Input Control](#).
    - ii. If a translation of description does not exist in the login language but a fallback language exists, the description in the fallback language is shown as a placeholder in [Description Input Control](#).
  - d. When editing is complete, click [Save](#).
4. To maintain translations for both a tracked process description and fields' label in a tracked process:
  - a. Click [Translate](#) button above the *User Model Fields* table
  - b. In the pop-up dialog, a translation of each field is displayed in one line. The first line is a description of this tracked process. Subsequent lines are user fields' label in this tracked process. In the filter, you can select which language needs to be modified. For example, if you select German in the [Language Drop Down List](#), a new column is added to the table below.
  - c. Add translation in the column of special line.
  - d. Click the [OK](#) button to save your changes.

## 5 DEPLOYED MODELS

When a model is deployed, two additional views become available:

- **Runtime View:** displays deployment information such as API details and deployment history log
- **Deployed View:** displays the deployed version of the model details

### 5.1 Runtime View

When a model is deployed, you can see the runtime information of the deployed model. The [Runtime](#) view contains the following information:

- Write Service API
- Read Service API

#### Procedure

1. On the *Model Details* page, click the [Runtime](#) button on the top right.
2. If you want the API of this model not to be exposed for consumption, click the [Deactivate](#) action. When you deactivate the model, its status changes to [Inactive](#).
3. If you want to expose the API for consumption, click the [Activate](#) action. When you activate the model, its status changes to [Active](#).

### 5.2 Write Service

#### Procedure

1. In the [Runtime](#) view, select the [Write Service](#) section to see the Write Service API details.
2. The [Write Service](#) section is displayed in the following order:
  - Tracked process types appear in alphabetical order.
  - ◇ In each tracked process type, the tracked process event is displayed first and followed by user-defined events in alphabetical order.
  - Core engine events appear at the end in alphabetical order.
3. The [Write Service](#) section contains the following information and functions that are offered to consume the Write Service API:
  - **API Endpoint:** allows you to test the API against the data available in the tenant account.
  - **Resources:** a list of the resources available for the API.
  - **API operations:** HTTP POST method can be used to all these resources.
  - **Try it out:** consume and test the selected API. You can view the response of the API in JSON format in the Response body section.
  - **Schemas:** The schema for the resources.

#### 5.2.1 Trying Out API

#### Procedure

1. In the [Servers](#) drop down, you can see the API endpoint for the [Try it out](#) function.
2. In the API documentation, choose the required resource and its HTTP method. To test the API, click [Try it out](#).
3. Enter the request payload in JSON format in the [Request body](#) text box.
4. Click [Execute](#) to execute the request.
5. In the [Server response](#) section, it shows the response with the endpoint.

### 5.3 Read Service

Please refer to the *Interface Reference Guide* and the interface:  
GTT\_V2\_OData\_Query (Tracked Process/Event)

## 5.4 Deployed View

### Context

After a model has been deployed, you can see its deployed version in the [Deployed](#) view.

### Procedure

1. On the *Model Details* page, click the [Deployed](#) button on the top right.
2. The [Deployed](#) view has the same layout and content as the [Draft](#) view. The difference is the [Deployed](#) view is read-only because you cannot modify the content of the deployed version of the model.

## 6 MORE TASKS


On the top right of *Model Details* page, there is an [Additional Options](#) icon  that contains the following actions.

### 6.1 Activate/Deactivate the Model

#### Context

When a model is deployed, you can activate/deactivate the model to expose/conceal the Write Service and Read Service APIs for consumption.

#### Procedure


1. Launch the MM app. Select a deployed model.
2. On the *Model Details* page, switch to the [Runtime](#) or [Deployed](#) view.
3. Click the [Additional Options](#) icon  and select [Activate](#) or [Deactivate](#).
4. Click [OK](#) in the confirmation box.
5. The deployed version of the model is then active/inactive.
  - If it's active, you can read the model metadata and business data, and write the business data.
  - If it's inactive, you can only read the model metadata.

### 6.2 Delete Business Data of the Model

#### Context

When a model is deployed, you can delete the business data of the logical system you specified, including the tracked processes with corresponding events.

#### Procedure

1. Launch the MM app. Select a deployed model.
2. On the *Model Details* page, switch to the [Runtime](#) or [Deployed](#) view.
3. Click the [Additional Options](#) icon  and select [Delete Business Data](#).
4. Select [All](#) or [Specified Logical System](#). If you choose the latter, enter the logical system name.
5. Click [Delete](#).





## 6.3 Export Business Data of the Model

### Context

When a model is deployed, you can export the business data of the model, including the tracked processes with corresponding events, and download the files in the ZIP format.

To use this function, you must have the `DataModelAdminTemplate` role to your role collection. For more, see the “[Available Roles for Solution Owners](#)” section in the *Administration Guide for Solution Owners*.

### Procedure

1. Launch the MM app. Select a deployed model.
2. On the *Model Details* page, switch to the [Runtime](#) or [Deployed](#) view.
3. Click the [Additional Options](#) icon  and select [Export Business Data](#).
4. In the pop-up, specify the date range for the data you wish to export. This date corresponds to the tracked process creation date. The duration between the start date and the end date cannot exceed 365 days.
5. Click [Export](#). A request is then generated under the [Exported Files](#) section. An icon is displayed on the right to show its processing status:
  - The progress icon  means the request is being processed.
  - The completion icon  means the request is successfully processed.
  - The warning icon  means the request fails to be processed. You must generate a new export request.
6. Once the request is completed, expand the request. You will see a list of the ZIP files that you can download. Each file represents a tracked process and its events. The file name contains the tracked process name and the date and time when it was created. The maximum file size is 100 MB. If the data for a tracked process exceeds this limit, a new file will be created.

Note: The export requests will be automatically deleted after 7 days.

7. Select a file that you want to download.
8. Click [Download](#). The file will then be saved to your local device. When you extract the file, you'll see a collection of JSON files in the resulting folder.


## 6.4 Enable/Disable the Model Cache

### Context

When a model is deployed, you can enable/disable the model cache. The default is *Enabled*. If you change the model cache switch from *Enabled* to *Disabled* or from *Disabled* to *Enabled*, the change needs at most 15 minutes to take effect.

**Attention:** In production tenants, *enabling the model cache* is strongly recommended to ensure optimal message processing performance. Disabling the model cache in production tenants significantly slows down the message processing and could lead to a great number of pending messages in the Manage Message Logs app.

### Procedure

1. Launch the MM app. Select a deployed model.
2. On the *Model Details* page, switch to the [Runtime](#) or [Deployed](#) view.
3. Click the [Additional Options](#) icon  and select [Enable/Disable Model Cache](#).
4. Click [OK](#) in the confirmation box.
5. The model cache is then enabled/disabled.
  - If it is *Enabled*, when you make any changes to the model and redeploy it, the model cache in the runtime needs at most 15 minutes to expire. New changes will take effect thereafter.
  - If it is *Disabled*, when you make any changes to the model and redeploy it, new changes will take effect at once.



## 6.5 Enable/Disable Event to Action Log of the Model

### Context


When a model is deployed, you can enable/disable the logging of certain items related to your event-to-action script. These items are recorded in message logs and event logs. They include:

- the “[Event to Action Log](#)” statements
- the payload sent to the tracking API through the function [sendTrackingRequest](#)
- the payload sent to SAP Transportation Management through the function [forwardEventToTM](#).

The default is *Disabled/Off*.

Once enabled, the system will start logging the above-mentioned items within 15 minutes. You can see the logs on the *Correlation Details* page of the [Manage Message Logs app](#) and the [Manage Event Logs app](#). For more information, see *Guide for Message Log Administrators*.

### Procedure


1. Launch the MM app. Select a deployed model.
2. On the *Model Details* page, switch to the [Runtime](#) or [Deployed](#) view.
3. Click the [Additional Options](#) icon  and select [Enable/Disable Event to Action Log](#).
4. Click [OK](#) in the confirmation box.
5. The Event to Action Log is then enabled/disabled.

## 6.6 View the Model History

### Context

When a model is deleted, deployed, or meets deployment/deletion error, you can see the history details about the model.

### Procedure

1. On the *Model Details* page, click the [Additional Options](#) icon  on the top right and select [History](#).
2. The history log is displayed in chronological descending order by the action time.
3. The history log entries are grouped by weeks. Click the group title to collapse/expand the group.
4. The history log entry contains the activity time and description.

## 7 CHECK PROCESSES AND EVENTS

This chapter is concerned with the *Check Processes and Events (CPE)* app.

### 7.1 About the CPE App

*Check Processes and Events (CPE)* displays a tracked process with its event data based on a deployed GTT metadata model. The app

- is used to display all the tracked processes that have been deployed, to test the models, and to check the data
- displays details of all its events for a selected tracked process
- displays the data for standard and user-defined fields in the model.

When you launch the *CPE* app, the *Process Type Overview* page is displayed that lists all active process types with high-level information about each. Click a specific process type to see its data details on the [Process List](#) page.

#### 7.1.1 In-app Help

When you launch the *CPE* app, you can turn on in-app help that provides on-screen explanations of key fields and areas on the screen.

To turn on in-app help:

- On the top right on the screen, click the [?](#) button
- the *Help Topics* Panel appears on the right

Once in-app help is turned on, you can:

- search the displayed Help topics for text you type
- click hot spots or bubbles to see help text on that topic
- hide the *Help Topics* Panel by clicking the *Hide (>>)* button on the bottom right
- toggle *Help* off

#### 7.1.2 Filters

You can apply one or more filters to reduce the number of process types displayed. You can also use value proposal search in selected filters. When you start typing, matching elements are displayed, and you can select them.

By default, the following filters are shown:

- [Process Type](#): the type of a tracked process.
- [Model Namespace](#): the full unique technical name of the model.
- [Last Deployed At](#): the range of start to end date that the model was last deployed.

When you have completed your filter selection, click [Go](#).

You have the following options to alter how filters are displayed:

- You can hide the filters by clicking the [^](#) button.
- You can pin the filters to the top of the screen by clicking the pin button.
- You can change the default filters, by clicking [Adapt Filters](#). Here, you can save your own views.

### 7.1.3 Views

You can select a view, also called a variant. The view determines the number of process types listed on the [Process List](#) page. There are two types, either:

- Standard: unfiltered so displays all active process types or
- one of the customized variants, if any have been created. These include some filters that may reduce the number of process types displayed.

After you select your view, its name is displayed on-screen. Click [Go](#) to display the corresponding process types.

An asterisk (\*) is displayed after the view name whenever any additional filter selection(s) is made. You can save your selection to create your own view.

To create your own view:

- specify one or more filters
- click the arrow to the right of the view
- enter a name for your view (case sensitive)
- You can also choose: *Set as Default* or *Apply Automatically*
- Save your view

By default, four columns are shown: *Process Type*, *Model Namespace*, *Description* and *Last Deployed at*.

You can change the default columns with the [Settings](#) icon to display the *View Settings* screen.

Note that you can also customize the table settings on the [Process Details](#) page and further pages and save the customized table as a variant.


### 7.1.4 Process List

From the *Process Type Overview* page, click a specific process type to see data details of that process type on the *Process List* page.

By default, there are five filters displayed:

1. [Alternative Key](#): the unique identifier of the tracked process in SAP Business Network for Logistics. It follows the pattern: scheme:party:system:type:id
  - Scheme: "xri://sap.com/id", a fixed value
  - Party: LBN ID of the business partner
  - System: name of the logical system that the tracked process comes from
  - Type: business or document type of the tracked process
  - ID: tracking ID of the tracked process
2. [Tracking ID](#): the document number of the tracked process
3. [Logical System](#): an application system in which the applications are coordinated to work in one common database, and from which the tracked process comes from
4. [Lifecycle Status](#): the lifecycle status of the tracked process. Select from the drop-down list one or more of:
  - [Business Active](#): the default value when a tracked process is created. It indicates that the relevant data of the tracked process is used in the ongoing business.
  - [End of Business](#): defines the end of active business and the start of residence time and retention period. When one of the last planned events is reported, the lifecycle status of the tracked process changes to "End of Business" automatically.

5. **Process Status**: the actual status of a tracked process, decided by the status of the last planned event whose status is not “As Planned”. Select from the drop-down list one or more of:
- **As Planned**: the initial status of a tracked process, also the status when the last planned event whose status is not “As Planned” is reported on time
  - **Delayed**: the status when the last planned event whose status is not “As Planned” is expected to be late
  - **Early**: the status when the last planned event whose status is not “As Planned” is early reported
  - **Late**: the status when the last planned event whose status is not “As Planned” is late reported
  - **Overdue**: the status when the last planned event whose status is not “As Planned” is overdue. Overdue indicates that the current timestamp exceeds the latest planned technical timestamp.

You can use value proposal search in selected filters by clicking the icon . The case insensitive search is supported for the following conditions:

- “contains”
- “starts with”
- “ends with”.

When you start typing, matching elements are displayed, and you can select them.

To change the default filters, click **Adapt Filters**. More filters are provided, such as:

- **Estimated Status**: the estimated status of a tracked process, decided by the estimated status of the unreported planned event that is next to the last reported planned event
- **ID**: the technical key of a tracked process
- **Party ID**: the LBN ID of a solution owner or a data contributor
- **Tracking ID Type**: the equivalent to the tracked process type

In the table of processes, you can change the default columns with the **Settings** button to display the *View Settings* screen.

Select a process to go to the next page: *Process Details*.

### 7.1.5 Process Details

There are four tabs displayed:

**1. Information:** divided into the following selectable parts:

- Core: core information that is predefined by the GTT system in the model. To see more information, click [Show More](#). After that, click [Show Less](#) to hide the extra information.
- Standard: standard information that is predefined in the GTT standard model.
- User-defined: defined by the user in the model
- Tracking IDs: core composition with table type that is predefined by the GTT system in the model. You can change the default columns and sorting with the [Settings](#) button. You can drill down further for more information.
- External Tracking IDs: core composition with table type that is predefined by the GTT system in the model. You can change the default columns and sorting with the [Settings](#) button. You can drill down further for more information.
- One-Time Locations: core composition with table type that is predefined by the GTT system in the model.
  - You can use one-time locations for planned events without maintaining the locations in the location master data.
  - Location geo coordinates will be automatically calculated based on the one-time location information provided, if they're not maintained.
  - The field "timeZone" supports the time zone IDs that are defined by IANA Time Zone Database. When setting values for the "timeZone" and "timeZoneld" fields, you should only provide a value for one of them. If you maintain the value of the parameter `GTT_OTL_TIMEZONE` in the Tracked Process EM IDOC, it will be automatically converted to the corresponding time zone ID. For more information on the mentioned IDOC, see *Interface Reference Guide*.

You can change the default columns and sorting with the [Settings](#) button. You can also drill down further for more information.
- Standard Composition Names: table-typed fields predefined in the GTT standard model
- User-defined Composition Names: table-typed fields defined by the user in the model. You can change the default columns and sorting with the [Settings](#) button. You can drill down further for more information.

### 2. Planned Events

The *Planned Events* table displays all the planned event data for the process in multiple columns including *ID*, *Event Type*, *Location Alternative Key*, *Event Match Key*, *Event Status*, *Planned Business Timestamp*, and *Actual Business Timestamp*.

Note: The *Actual Business Timestamp* is for the planned event's last correlated event\*.

\*last correlated event: When an actual event is matched with a planned event successfully and the status of this planned event is calculated based on this actual event, then this actual event is considered as the last correlated event of this planned event.

You can change the default columns and sorting with the [Settings](#) button. You can also drill down further for more information.

### 3. Other Events

- GTTUpdatePlanEvent
- Unplanned events - displays all the actual event data that has been reported without any reference to a planned event.

You can change the default columns and sorting with the [Settings](#) button. You can also drill down further for more information.

#### 4. Attachments

Displayed only when you create a user model field with its type as [Attachments](#) in the [Tracked Process](#) tab of the MM app.

A [Refresh](#) button is displayed on the top right. Click the button to update the data shown.

##### 7.1.6 Planned Event Details

On the *Process Details* page, select a planned event from the *Planned Events* table to see the *Planned Event Details* page.

There are two tabs:

###### 1. Planned Data

Information about the planned event including the core fields such as *ID*, *Event Status*, *Planned Timestamp*, *Geo Position*, standard fields and the user-defined fields. To see more information, click [Show More](#). After that, click [Show Less](#) to hide the extra information.

###### 2. Actual Data

Only displayed if an actual event is reported with reference to this planned event. The information is held in a table. You can change the default columns and sorting with the [Settings](#) button.

Select an event to see the *Event Details* page.

A [Refresh](#) button is displayed on the top right. Click the button to update the data shown.

##### 7.1.7 Event Details

On the *Process Details* page, select an event from the *Other Events* table to see the *Event Details* page. You can also see this page when you select an event from the *Actual Data* table on the *Planned Event Details* page.

There are six tabs:

1. **Information:** with details for the core fields such as *ID*, *Actual Timestamps*, *Event Reason Text* and *Sender Party ID*, standard fields, and the user-defined fields. To see more information, click [Show More](#). After that, click [Show Less](#) to hide the extra information.
2. **References:** displayed in a table with columns such as *ID*, *Event ID*, and *Reference Type*. You can change the default columns and sorting with the [Settings](#) button. You can also drill down further for more information.
3. **Event Processes:** displayed in a table with columns such as *Process*, *Planned Event Location Alternative Key*, and *Correlation Type*. You can change the default columns and sorting with the [Settings](#) button.  
You can click the link to a process in a row to see its *Process Details* page.  
You can also drill down further for more information.
4. **Estimated Timestamps:** displayed in a table with columns such as *ID*, *Event ID*, *Event Type*, *Location Alternative Key*, *Event Match Key* and *Estimated Timestamp*. You can change the default columns and sorting with the [Settings](#) button.
5. **Standard Composition Names:** table-typed fields predefined in the GTT standard model

6. **User-defined Composition Names:** table-typed fields defined by the user in the model. The information is displayed in a table with columns such as *Stop ID*, *Planned Arrival At*, and *Location Time Zone*. You can change the default columns and sorting with the [Settings](#) button. You can also drill down further for more information.

## 7.2 Check Active Process Types

### Context

To check the process types in all active models, do the following:

### Procedure

1. Launch the CPE app.
2. The *Process Type Overview* page displays the list of process types for all the active models.
3. As previously explained, you can reduce the number of process types displayed by setting one or more [filters](#) or using a [View](#).

### 7.2.1 Check Active Processes

### Context

To check active processes, do the following:

### Procedure

1. Launch the CPE app.
2. The *Process Type Overview* page displays the list of process types for all the active models.
3. Select a process type from the list and click it.
4. The [Process List](#) page displays a table of all active processes.

### 7.2.2 View Tracked Process Details

### Context

To view the details of a tracked process, do the following:

### Procedure

1. Launch the CPE app.
2. The *Process Type Overview* page displays the list of process types for all the active models.
3. Select a process type from the list and click it.
4. The [Process List](#) page displays a table of all active processes.
5. Select a process from the list and click it.
6. The [Process Details](#) page shows the information of the selected process arranged by:
  - *Information*
  - *Planned Events*
  - *Other Events*



### **7.2.3 View Planned Event Details**

#### **Context**

To view the details of a planned event, do the following:

#### **Procedure**

1. Launch the CPE app.
2. The *Process Type Overview* page displays the list of process types for all the active models.
3. Select a process type from the list and click it.
4. The *Process List* page displays a table of all active processes.
5. Select a process from the list and click it.
6. The *Process Details* page shows the information of the selected process arranged by:
  - *Information*
  - *Planned Events*
  - *Other Events*
7. Click *Planned Events* to see the details.

### **7.2.4 View Actual Event Details**

#### **Context**

To view the details of an actual event, do the following:

#### **Procedure**

1. Launch the CPE app.
2. The *Process Type Overview* page displays the list of process types for all the active models.
3. Select a process type from the list and click it.
4. The *Process List* page displays a table of all active processes.
5. Select a process from the list and click it.
6. The *Process Details* page shows the information of the selected process arranged by:
  - *Information*
  - *Planned Events*
  - *Other Events*
8. Click *Other Events* to see the details of actual events without reference to any planned event.
9. To check the actual data with reference to the planned event, on the *Planned Events* tab, click a planned event in the *Planned Events* table to display the *Planned Event Details* page. Then on the *Actual Data* tab, you can drill down to view details including *Information*, *References*, *Event Processes*, *User-defined Compositions*.

## 8 APPENDIX: EVENT TO ACTION SCRIPT

An Event to Action script allows you to incorporate logic (business rules) into your model. This section explains about the structure and syntax of the script.

To add your script to a model, see the section [Event to Action](#).

To know more about Reminder service, see *Reminder Service for Version 2*.

### Lexical Structure

The lexical structure of a programming language is the set of elementary rules that specifies how you write programs in that language. It is the lowest-level syntax of a language; it specifies such things as what variable names look like, the delimiter characters for comments, and how one program statement is separated from the next. This section documents the lexical structure of the *Event to Action* script.

#### 1.1 Character Set

##### 1.1.1 Case Sensitivity

This script is a case-sensitive language. This means that language keywords, variables, function names, and other *identifiers* must always be typed with a consistent capitalization of letters. The `while` keyword, for example, must be typed “while,” not “While” or “WHILE.” Similarly, `online`, `Online`, `OnLine`, and `ONLINE` are four distinct variable names.

##### 1.1.2 Whitespace

This script recognizes the following characters as whitespace: space(\u0020), tab(\t), carriage return: (\r), line feed (\n).

#### 1.2 Comments

This script supports the following styles of comments:

- Any text between a `//` and the end of a line is treated as a comment and is ignored.
- Any text between the characters `/*` and `*/` is also treated as a comment; these comments may span multiple lines but may not be nested.

#### 1.3 Literals

A literal is a data value that appears directly in a program. The following are all literals:

<code>12</code>	<code>// The Integer number</code>
<code>1.2</code>	<code>//The float number</code>
<code>"hello world"</code>	<code>//A string value</code>
<code>true</code>	<code>//A Boolean value</code>
<code>false</code>	<code>//The other Boolean value</code>
<code>null</code>	<code>//null value</code>
<code>[1,2,3,4,5]</code>	<code>//An array</code>

## 1.4 Identifiers and Reserved Words

An *identifier* is simply a name. Identifiers are used to name variables and functions and to provide labels for certain loops in script code. An identifier must begin with a letter, or an underscore (\_). Subsequent characters can be letters, digits, underscores. The following are all legal identifiers:

```
a
a_b_c
x11
_test
```

### 1.4.1 Reserved Words

The script reserves a number of identifiers as the keywords of the language itself. You cannot use these words as identifiers in your programs:

actualEvent	plannedEvent	trackedProcess		
break	function	return	case	do
if	switch	var	else	continue
false	while	finally	true	for

## 1.5 Semicolons

Like many programming languages, we use the semicolon (;) to separate statements from each other. This is important to make the meaning of your code clear: without a separator, the end of one statement might appear to be the beginning of the next, or vice versa.

## Values and Variables

### 2.1 String Literals

To include a string literally in a program, simply enclose the characters of the string within a matched pair of double quotes ("). The maximum size for a string is 10M. Here are examples of string literals:

```
"testing"
```

```
"π is the ratio of a circle's circumference to its diameter"
```

Function **length()** returns the length of a specified string. For example:

```
var str = "circle";
```

```
str.length(); // returns 6
```

Function **search()** searches for a specified string value and returns the position of the match. If no matches are found, the function returns -1. For example:

```
var str = "π is the ratio of a circle's circumference to its diameter";
```

```
str.search("is"); // returns 2
```

```
str.search("t"); // returns 5
```

Function **substring()** extracts the characters in a string from the start character to the end character and returns a new substring. To use this function, the following points should be noticed:

- The start index is mandatory. If the end index is not specified, the function extracts from the start character to the end of the string.
- The start character is included in the substring, but the end one is not.
- Only integers are input as indices. Subtractive integers are recognized as 0.
- The index 0 suggests the first character.
- If the start index is greater than the end index, such as *str.substring(4, 1)*, the indices will automatically swap places to *str.substring(1, 4)*.

Here are examples of using this function:

```
var str = "π is the ratio of a circle's circumference to its diameter";
```

```
str.substring(2, 4); // returns "is"
```

```
str.substring(5); // returns "the ratio of a circle's circumference to its diameter"
```

### 2.2 Boolean Values

A boolean value represents true or false, on or off, yes or no. There are only two possible values of this type. The reserved words **true** and **false** evaluate to these two values.

### 2.3 null Value

null is a language keyword that evaluates to a special value that is usually used to indicate the absence of a value.

### 2.4 Object Value

An object is a composite value: it aggregates multiple values (primitive values or other objects) and allows you to store and retrieve those values by name. An object is an unordered collection of properties, each of which has a name and a value. Property names are strings, so we can say that objects map strings to values. This string-to-value mapping goes by various names. The maximum size for an object is 100M.

Function **containsProperty()** checks if an object contains a property. The syntax of this function is:

```
object.containsProperty("propertyName")
```

For example: *o.containsProperty("x")*

## 2.5 Array Value

An array is an ordered collection of values. Each value is called an element, and each element has a numeric position in the array, known as its index. An array element may be of any type, and different elements of the same array may be of different types. Array elements may even be objects or other arrays, which allows you to create complex data structures, such as arrays of objects and arrays of arrays.

There are two functions for an array value:

1. `array.size()`, return the size of the array.
2. `array.push(x)`, add element `x` to the array as the last element.

## 2.6 Numbers

Number value can be integer or float. A number value can be transformed to String like this:

*`num.toString()`*

## 2.7 Timestamp Value

There are four functions for a timestamp value:

1. `timestamp.minusDays(x)`, subtract the number of days (`x`) in the timestamp.
2. `timestamp.plusDays(x)`, add the number of days (`x`) in the timestamp.
3. `timestamp.minusMinutes(x)`, subtract the number of minutes (`x`) in the timestamp.
4. `timestamp.plusMinutes(x)`, add the number of minutes (`x`) in the timestamp.

## Expressions and Operators

### 3.1 Property Access Expressions

A property access expression evaluates to the value of an object property or an array element.

Syntax for Object value property access:

*expression . identifier*

Here are some examples:

```
var o = toObject("{x:1,y:{z:3}}");           //An example object
o.x;                                           //=>1
o.y.z;                                         //=>3
```

Syntax for Array value property access:

*expression [ expression ]*

Here are some examples:

```
var o = toObject("{x:1,y:{z:3}}");           //An example object
var a = [o,4,[5,6]];                          // An example array that contains the object
a[0].x;                                         //=>1
```

### 3.2 Invocation Expressions

An invocation expression is syntax for calling (or executing) a function or method. It starts with a function expression that identifies the function to be called. The function expression is followed by an open parenthesis, a comma-separated list of zero or more argument expressions, and a close parenthesis.

Some examples:

```
toObject("(");           //toObject is the function expression; "(" is the argument expression
var uuid = toUUID("00000000-0000-0000-0000-000000000000");
var date = toDate("2010-05-04");
var timestamp = toTimestamp("2013-06-25T16:22:52.966Z");
```

### 3.3 Operators

Operators are used for arithmetic expressions, comparison expressions, logical expressions, assignment expressions, and more. Here are all allowed operators:

Operator	Operation	Example
+	Positive number	+1
-	Negate number	-1
+, -	Add, subtract	1+2 1-2
<, <=, >, >=	Compare	a>=b
==	Test for equality	a==b
!=	Test for inequality	a!=b
^	Power	2^3 == 8
&&	Compute logical AND	true && false
	Compute logical OR	true    false
!	Invert boolean value	!true
=	assignment	a=1



## Statements

### 4.1 Declaration Statements

The “var” statement declares a variable. Here is the syntax:

```
var name [ = value]
```

### 4.2 Return Statement

The return statement terminates the execution of the code block immediately.

### 4.3 Conditions

Conditional statements execute or skip other statements depending on the value of a specified expression. These statements are the decision points of your code, and they are also sometimes known as “branches.”

The “if” statement is the fundamental control statement that allows script to make decisions, or, more precisely, to execute statements conditionally. This statement has 3 forms. The first is:

```
if (expression)  
statement
```

The second form of the “if” statement introduces an else clause that is executed when expression is false. Its syntax is:

```
if (expression)  
statement1  
else  
statement2
```

The third form’s syntax is:

```
if (expression)  
statement1  
else if(expression2)  
statement2  
else  
statement2
```

## 4.4 Loops

### 4.4.1 while

The script supports “while” statement for loop. It has the following syntax:

```
while (expression)
statement
```

To execute a “while” statement, the interpreter first evaluates the expression.

- If the value of the expression is false, then the interpreter skips over the statement that serves as the loop body and moves on to the next statement in the program.
- If the expression is true, the interpreter executes the statement and repeats, jumping back to the top of the loop and evaluating the expression again.
- The *continue* statement terminates the execution of the statements in the current iteration of the current loop and continues the execution of the loop with the next iteration.
- The *break* statement terminates the current loop statement and transfers the program control to the statement following the terminated statement.

Example:

```
var i = 0;
var n = 0;

while (i < 5) {
  i = i+1;
  if (i == 3) {
    continue;
  }
  n = n+i;
  if (n > 8) {
    break;
  }
}
```

## 4.5 Event to Action Log

The “Event to Action Log” statement begins a code line with “?” and is followed by a variable.

For example:

```
var test = "gtttest5";
?test ;
```

To see your event-to-action logs, you should do the following in the MM app first:

1. Deploy the model
2. [Enable the Event to Action Log.](#)

After the message processing, the logs are displayed on the [Correlation Details](#) page -> [Process History](#) tab -> [View Logs](#) button -> [Event to Action Internal](#) phase in the *Manage Message Logs* app and *Manage Event Logs* app.

Note:

- For each process event or actual event, at most 100 event-to-action logs are saved. Total log size exceeding 2 MB is truncated.
- The system keeps the logs for only two hours and deletes the out-of-date logs once an hour.

#### 4.6 User-defined Function

You can define your own functions. It must follow the syntax below:

```
function functionName(param1, param2, param3.....) {  
    statement  
}
```

Note that you can't use system embedded functions in your user-defined functions.

*Example:*

```
function sample(a, b){  
    ?a;  
    ?b;  
    return a+b;  
}  
var c=sample(1,3); // c is 4 after execution
```

#### Samples

Reminder service can be enabled by Event to Action. For details, search for the document *Reminder Service for Version 2* at [help.sap.com/gtt](https://help.sap.com/gtt) and see the chapter "Implement the Script".

## System Embedded Functions and Constants

There are some pre-defined system functions and constants that you can invoke directly in the script.

### 5.1 System Embedded Constants

#### 5.1.1 **trackedProcess**

The constant **trackedProcess** is an object of the tracked process.

The object structure example please refer to sub-section **6.1**.

Function **getFinalPlannedEvents()** under **trackedProcess** object returns a list of events that are flagged as final event.

For example:

```
trackedProcess.getFinalPlannedEvents();
```

#### 5.1.2 **actualEvent**

The constant **actualEvent** is an object of the actual event received.

The object structure example please refer to sub-section **6.2**.

#### 5.1.3 **plannedEvent**

The constant **plannedEvent** is an object of the planned event.

The object structure example please refer to sub-section **6.3**.

Function **isReported()** under **plannedEvent** object returns a Boolean value true or false.

For example:

```
plannedEvent.isReported();
```

#### 5.1.4 **modelNameSpace**

The constant **modelNameSpace** is a string value of the current model namespace.

For example:

```
if(modelNameSpace + ".SalesOrder.SalesOrderEvent" == actualEvent.eventType) {  
...  
}
```

## 5.2 System Embedded Functions

System embedded functions are divided into two groups:

- Event to Action Internal functions: are used within SAP Business Network Global Track and Trace.
- Event to Action External functions: interact with third-party systems in their operations.

### 5.2.1 Event to Action Internal Functions

- **toTimestamp**

Function **toTimestamp** parses a timestamp in string format into TimestampValue.

```
/**transform a string to Timestamp value
 * @param timeString A timestamp in string format like "2013-06-25T16:22:52.966Z"
 * @returns TimestampValue
 */
toTimestamp(timeString)
```

- **toDate**

Function **toDate** parses a date in string format into DateValue.

```
/**transform a string to Date value
 * @param dateString A date in string format like "2019-05-04"
 * @returns DateValue
 */
toDate(dateString)
```

- **toUUID**

Function **toUUID** parses a UUID in string format into UUIDValue.

```
/**transform a string to UUID value
 * @param udiString A uuid string like "00000000-0000-0000-0000-000000000000"
 * @returns UUIDValue
 */
toUUID(udiString)
```

- **toObject**

Function **toObject** parses a JSON string into ObjectValue.

```
/**generate a JSON string to object value
 * @param jsonString A JSON string
 * @returns ObjectValue
 */
toObject(jsonString)
```

- **toString**

Function **toString** parses the value of a timestamp, date, or object type field into a string. The format of timestamp and date string is fixed as shown in the examples below.

```
/**transform a timestamp or date value to a string
 * @returns String
 */
date.toString()
time.toString()
object.toString()
```

For example:

```
var date = toDate("2022-03-19");
var string = date.toString();// Value of string is: 2022-03-19

var time = toTimestamp("2022-03-19T16:22:52.966Z");
var string = time.toString();// Value of string is: 2022-03-19T16:22:52.966Z
```

## Guide for Model Administrators

```
var object = toObject("{}");
var string = object.toString();// Value of string is: {}
```

### • updateTPProperties

Function **updateTPProperties** updates properties of a tracked process.

```
/**update properties of a tracked process
 * @param trackedProcess trackedProcess Object
 * @param properties Object contains properties to be updated in the tracked process.
 * @returns null
 */
updateTPProperties(trackedProcess, properties)
```

For example:

```
var obj = toObject("{}");
var a = toObject("{\n" +
    "                \"batchNumber\": null,\n" +
    "                \"deliveryItem\": \"000020\",\n" +
    "                \"deliveryItemRef\": \"102\",\n" +
    "            }");

var b = toObject("{\n" +
    "                \"batchNumber\": null,\n" +
    "                \"deliveryItem\": \"000030\",\n" +
    "                \"deliveryItemRef\": \"102\",\n" +
    "            }");

obj.DeliveryItem = [a, b];
updateTPProperties(trackedProcess, obj);
```

### • updatePlan

After you change the timestamp of the planned event in the event-to-action script, you can use function **updatePlan** to recalculate the planned event status and the tracked process status, and reset overdue detection.

```
/**recalculate planned event status and tracked process status, reset overdue detection
 * @param plannedEventIdArray An array of planned event ID needed to be recalculated
 * @returns null
 */
trackedProcess.updatePlan([id1, id2, ...])
```

For example:

```
var obj = toObject("{}");
var plannedEvents = trackedProcess.plannedEvents;
plannedEvents[0].plannedBusinessTimestamp = toTimestamp("2021-06-16T11:00:00.213+01:00");
plannedEvents[0].plannedBizTsEarliest = toTimestamp("2021-06-16T10:00:00.213+01:00");
plannedEvents[0].plannedBizTsLatest = toTimestamp("2021-06-16T12:00:00.213+01:00");
var id = plannedEvents[0].id;
obj.plannedEvents = plannedEvents;
updateTPProperties(trackedProcess, obj);
trackedProcess.updatePlan([id]);
```

- **getLocationByAltKey**

Function **getLocationByAltKey** gets the location master data you maintained in the Manage Locations app via the location alternative key.

```
/**get location master data
 * @param location Altkey A string
 * @returns ObjectValue of location data
 */
getLocationByAltKey(locationAltkey)
```

**Properties:**

<i>StreetName</i>	<i>LocationDescription</i>
<i>HouseNumberSupplementText</i>	<i>Number: Phone number</i>
<i>HouseNumber</i>	<i>RegionName</i>
<i>RegionCode</i>	<i>NumberExtension: Phone extension number</i>
<i>CountryName</i>	<i>CountryCode</i>
<i>IataAirportCode</i>	<i>LocationTypeCode</i>
<i>AddressTimeZone</i>	<i>LocationTypeDescription</i>
<i>PostalCode</i>	<i>SourceSystem</i>
<i>Latitude</i>	<i>SourceSystemType</i>
<i>Longitude</i>	<i>ExternalId</i>
<i>CityName</i>	<i>ObjectTypeCode</i>
<i>EmailAddress</i>	<i>ObjectTypeDescription</i>
<i>Unlocode</i>	<i>LocationTypeCode</i>
<i>SourceUniversalObjectId: Source System/External ID</i>	<i>LocationTypeDescription</i>

**Examples:**

**Using getLocationByAltKey**

```
var loc = getLocationByAltKey("xri://sap.com/id:LBN#10013165:01:Location:LogisticLocation:02");
```

**Response**

```
{
  "StreetName": null,
  "HouseNumber": null,
  "RegionCode": null,
  "CountryName": null,
  "ObjectTypeCode": "LogisticLocation",
  "PostalCode": null,
  "Latitude": null,
  "SourceSystemType": "SCM",
  "LocationDescription": "alex test logistic location",
  "Number": null,
  "RegionName": null,
  "LocationTypeCode": "1001",
  "CountryCode": null,
  "HouseNumberSupplementText": null,
  "ObjectTypeDescription": "Logistic Location",
  "IataAirportCode": "",
  "LocationTypeDescription": "Production Plant",
  "ExternalId": "001",
  "AddressTimeZone": null,
  "CityName": null,
  "Longitude": null,
  "EmailAddress": null,
  "Unlocode": "",
  "SourceUniversalObjectId":
"xri://sap.com/id:LBN#10013165:LOGISTICLOCATION:Location:LogisticLocation:001",
  "NumberExtension": null,
  "SourceSystem": "LOGISTICLOCATION"
}
```

**Using a property of Location**

```
loc.StreetName
```

- **getEventTypeInfo**

Function **getEventTypeInfo** gets the name, description, and fields of a given event type. If the Java locale string is provided, you can also get its description. This feature is available in the following languages:

<i>Language</i>	<i>Language Code</i>
Arabic	ar
Bulgarian	bg
Czech	cs
Danish	da
German	de
Greek	el
English	en
Spanish	es
Finnish	fi
French	fr
Croatian	hr
Hungarian	hu
Italian	it
Hebrew	iw
Japanese	ja
Korean	ko
Malay	ms
Dutch	nl
Norwegian	no
Polish	pl
Portuguese - Brazil	pt_BR
Romanian	ro
Russian	ru
Slovak	sk
Serbian	sr
Swedish	sv
Thai	th
Turkish	tr
Simplified Chinese	zh_CN
Traditional Chinese	zh_TW

If there is no text in the given language, the function returns the English text. If the event type doesn't exist, the function returns an empty string.

```

/**get event type info
 * @param Event type full name A string
 * @param language code Java locale string
 * @returns ObjectValue of event type info
 */
getEventTypeInfo(eventTypeFullName, languageCode)

```

#### Properties:

<i>descr</i>	<i>parent</i>
<i>elements</i>	<i>idocMapping</i>
<i>vpMapping</i>	<i>name</i>



### Examples:

#### Using `getEventTypeInfo`

```
var eventTypeInfo = getEventTypeInfo("com.gttsampleso01.gtt.app.test.ArriveEvent", "zh_CN")
```

#### Response

```
{
  "descr": "Arrival Event",
  "parent": {
    "target": "CoreModel.Event"
  },
  "elements": [{
    "name": "stringTest",
    "label": "String Test",
    "type": "string",
    "readable": true,
    "writable": true,
    "translation": {
      "info": {
        "textType": "LABEL",
        "translation": {
          "zh_CN": "字符串测试"
        }
      },
      "defaultText": "String Test",
      "textType": "LABEL",
      "fieldPath": "label",
      "contextPath": "elements/0",
      "label": {
        "textType": "LABEL",
        "translation": {
          "zh_CN": "字符串测试"
        }
      }
    },
    "length": 50
  }],
  "idocMapping": {
    "erpEventCode": "",
    "idoc": "",
    "fieldMapping": []
  },
  "vpMapping": {
    "lbnEventType": "",
    "fieldMapping": []
  },
  "name": "com.gttsampleso01.gtt.app.EventTypeInfoProcess.ArrivalEvent"
}
```

#### Using a property of Event Type Information

```
eventTypeInfo.descr
```

- **getCodeListText**

Function **getCodeListText** gets the text of the code of a given code list in a given language. This feature is available in the following languages:

<i>Language</i>	<i>Language Code</i>
Arabic	ar
Bulgarian	bg
Czech	cs
Danish	da
German	de
Greek	el
English	en
Spanish	es
Finnish	fi
French	fr
Croatian	hr
Hungarian	hu
Italian	it
Hebrew	iw
Japanese	ja
Korean	ko
Malay	ms
Dutch	nl
Norwegian	no
Polish	pl
Portuguese - Brazil	pt_BR
Romanian	ro
Russian	ru
Slovak	sk
Serbian	sr
Swedish	sv
Thai	th
Turkish	tr
Simplified Chinese	zh_CN
Traditional Chinese	zh_TW

If there is no text in the given language, the function returns the English text. If the code or code list doesn't exist, the function returns an empty string.

```
/** get the text of the code
 * @param codeType the name of the CodeList
 * @param codeValue the value of the code
 * @param languageCode Java locale string
 * @returns String
 */
getCodeListText(codeType, codeValue, languageCode)
```

Examples:

*Using getCodeListText*

```
// the variable processStatus is the text in Chinese of Code "AS_PLANNED" of the CodeList
"ProcessStatus"
var processStatus = getCodeListText("ProcessStatus", "AS_PLANNED", "zh_CN");
```

## • getBPByLBNID

Function **getBPByLBNID** gets business partner master data via LBN ID.

```
/**get business partner master data
 * @param LBNID A string
 * @returns ObjectValue of business partner master data
 */
getBPByLBNID(LBNID)
```

### Properties:

<i>SubDomain</i>	<i>TenantId</i>
<i>Email</i>	<i>BpName</i>
<i>Id: technical UUID</i>	<i>bpContactInfos (include ContactType, Email, Phone, FirstName, LastName)</i>
<i>LbnId</i>	<i>bpAccountDetails (include Status, GlobalAccExternalGUID, SubAccExternalGUID, BpAccId, SubscribedPlan, BpId, BpType, SubaccountId, SubscribedApp)</i>

### Examples:

#### Using getBPByLBNID

```
var bp = getBPByLBNID("LBN#10013165");
```

#### Response

```
{
  "SubDomain": "gtt-sample-so-01",
  "TenantId": "20d81d3a-1ff0-49d5-a737-ec98ae7cf587",
  "Email": null,
  "BpName": "Test Company",
  "Id": "D7964ECC-DE35-4F1E-9090-8D3755B67170",
  "bpContactInfos": [
    {
      "ContactType": "LBN_IT",
      "Email": "lily.lee@abc.com",
      "Phone": "12345678",
      "FirstName": "Lily",
      "LastName": "Lee"
    },
    {
      "ContactType": "LBN_LOGISTICS",
      "Email": "robert.swift@abc.com",
      "Phone": "13312345678",
      "FirstName": "Robert",
      "LastName": "Swift"
    }
  ],
  "bpAccountDetails": [
    {
      "Status": "Active",
      "GlobalAccExternalGUID": "iotDigitalScm",
      "SubAccExternalGUID": "gtt-sample-so-01",
      "BpAccId": "a6e6577b-61cb-4f5c-942c-dc5515c67d4d",
      "SubscribedPlan": "test",
      "BpId": "D7964ECC-DE35-4F1E-9090-8D3755B67170",
      "BpType": "GTT_SOLUTION_OWNER",
      "SubaccountId": "20d81d3a-1ff0-49d5-a737-ec98ae7cf587",
      "SubscribedApp": "lbn_gtt_core_int-app"
    },
    {
      "Status": "Active",
      "GlobalAccExternalGUID": "iotDigitalScm",
      "SubAccExternalGUID": "gtt-sample-so-01",
      "BpAccId": "e7fbb5ad-16a5-4803-90e4-afe8e7ca57ec",
      "SubscribedPlan": "test",
      "BpId": "D7964ECC-DE35-4F1E-9090-8D3755B67170",
      "BpType": "OTHERS",
      "SubaccountId": "20d81d3a-1ff0-49d5-a737-ec98ae7cf587",
      "SubscribedApp": "lbn-dev"
    }
  ]
}
```

## Guide for Model Administrators

```
    },  
    "LbnId": "LBN#10013165"  
  }  
}
```

### Using a property of Business Partner Master Data

```
bp.bpAccountDetails[0].status
```

*\*Note that [0] stands for the first business partner account.*

### • getTimeDifference

Function **getTimeDifference** calculates the time difference between two given timestamp values and returns the result in a specified time unit, such as seconds, minutes, hours, or days.

```
/**get time difference  
 * @param timestamp1 A timestamp value  
 * @param timestamp2 A timestamp value  
 * @param unit Use "seconds", "minutes", "hours", or "days". It's case insensitive.  
 * @returns Decimal Rounded to two decimal places using rounding rules. The result is a  
positive number if timestamp1 is later than timestamp2. The result is a negative number if  
timestamp1 is earlier than timestamp2.  
 */  
getTimeDifference(timestamp1, timestamp2, unit)
```

For example:

```
var timestamp1 = toTimestamp("2022-05-03T12:34:56.000Z");  
var timestamp2 = toTimestamp("2022-05-11T17:45:58.000Z");  
var unit1 = "days";  
var difference1 = getTimeDifference(timestamp1, timestamp2, unit1);  
// The timestamp difference1 by unit1 is: -8.22 days
```

- **generateDirectMailTemplate**

Function **generateDirectMailTemplate** defines an instance of the email template object.

```
/**generate email template
 * @returns Email template
 */
reminderService.generateDirectMailTemplate();

directMailTemp.templateId = "name of the email template";
directMailTemp.locale = "language code";
directMailTemp.recipients = ["email address list"];
directMailTemp.userdefinedPlaceholders = userDefPlaceholders;
directMailTemp.templateRecipientIgnore = true/false
```

For more information about the properties, refer to the section “Create Mail Object” in *Reminder Service for Version 2*.

For example:

```
var directMailTemp = reminderService.generateDirectMailTemplate();
directMailTemp.templateId = "invite";
directMailTemp.locale = "en";
directMailTemp.recipients = [jane@doe.com, "jack@ryan.com"];
directMailTemp.userdefinedPlaceholders = userDefPlaceholders;
directMailTemp.templateRecipientIgnore = false
```

- **post**

Function **post** triggers the process of sending the email.

```
/**send email
 * @returns null
 */
directMailTemp.post();
```

For example:

```
directMailTemp.post();
```

## Guide for Model Administrators

- **parseInt**

Function **parseInt** converts a string to an integer.

```
/** convert a string to an integer
 * @param string a string to be parsed
 * @returns an integer parsed from string
 */
```

For example:

```
var string = "12345678";
var parsedInteger = parseInt(string);
```

- **parseFloat**

Function **parseFloat** converts a string to a floating-point number, which is a number with a decimal point.

```
/** convert a string to a float
 * @param string a string to be parsed
 * @returns a float parsed from string
 */
```

For example:

```
var string = "12345678978987654321.98765432123456789";
var parsedFloat = parseFloat(string);
```

- **parseTPAltkey**

Function **parseTPAltkey** parses the alternative key of a tracked process or an event to get its properties including scheme, party, logical system, type, and ID.

```
/** parse the altkey of a tracked process or event
 * @param altkey A string of alternative key
 * @returns Object of parse result
 */
parseTPAltkey(altkey);
```

Properties:

<i>scheme</i> : "xri://sap.com/id"	<i>type</i> : business or document type of the tracked process
<i>party</i> : LBN ID of the business partner	<i>id</i> : tracking ID of the tracked process
<i>system</i> : name of the logical system that the tracked process comes from	

For example:

*Using parseTPAltkey*

```
var object =
  parseTPAltkey("xri://sap.com/id:LBN#10002424:QM7CLNT910:OUTBOUND DEL AUTO:1000004601");
?object;
```

*Response*

```
{ "scheme": "xri://sap.com/id", "logicalSystem": "QM7CLNT910", "id": "1000004601", "type": "OUTBOUND DE
L_AUTO", "party": "LBN#10002424" }
```

*Using a property of the tracked process alternative key*

```
var trackingId = object.id;
```

- **parseLocationAltKey**

Function **parseLocationAltKey** parses the alternative key of a location to get its properties including scheme, party, system, type, and ID.

```
/** parse the altkey of a location
 * @param altkey A string of alternative key
 * @returns Object of parse result
 */
parseLocationAltKey(altkey);
```

**Properties:**

<i>scheme</i> : "xri://sap.com/id"	<i>type</i> : object type of the location
<i>party</i> : LBN ID of the business partner	<i>id</i> : location ID
<i>system</i> : name of the system that the location comes from	

For example:

**Using parseLocationAltKey**

```
var pLocationAltKey = "xri://sap.com/id:LBN#10002424:Q8JCLNT774:Location:ShippingPoint:01";
var altkey = parseLocationAltKey(pLocationAltKey);
```

**Response**

```
{ "scheme": "xri://sap.com/id", "party": "LBN#10002424", "system": "Q8JCLNT774", "type": "ShippingPoint", "id": "01" }
```

**Using a property of the location alternative key**

```
var id = altkey.id;
?"ID:" + id;
```

### • **composeLocationAltKey**

Function **composeLocationAltKey** combines the properties including party, system, type, and ID to get the location alternative key.

```
/** compose the location altkey
 * @param party A string of the business partner's LBN ID
 * @param system A string of name of the system that the location comes from
 * @param type A string of location object type
 * @param id A string of location ID
 * @returns String of location alternative key
 */
composeLocationAltKey(party, system, type, id);
```

For example:

Using *composeLocationAltKey*

```
var cLocationAltKey = composeLocationAltKey("LBN#10002424", "Q8JCLNT774", "ShippingPoint",
"01");
?"composeLocationAltKey:" + cLocationAltKey;
```

Response

```
"xri://sap.com/id:LBN#10002424:Q8JCLNT774:Location:ShippingPoint:01"
```

### • **convertUTCToTimeZone**

Function **convertUTCToTimeZone** converts the UTC time to a specified time zone.

```
/** convert the UTC time to a specified time zone
 * @param timestamp A timestamp of the UTC time
 * @param timeZoneID A string used to identify the time zone. There are two types of
timeZoneID:
    • time offset: the number of hours or minutes a certain time zone is ahead of or
      behind UTC, GMT, or UT. Examples: "UTC-12:00", "GMT+2", "UT+02", "UTC+05:30:00".
    • time zone region: time zones defined by IANA Time Zone Database. Usually follow a
      "Area/Location" format, with "Area" being a continent or a large geographic
      region, and "Location" being a specific city or region within that area, such as
      "Europe/Paris".
```

If the timeZoneID is invalid, the system returns the timestamp as a string.

```
 * @returns A string in the "year-month-day hour:minute:second" format.
 */
convertUTCToTimeZone(timestamp, timeZoneID);
```

Examples:

*Converting the time from a string*

```
var timestamp = toTimestamp("2023-04-13T05:45:00Z");
var converted = convertUTCToTimeZone(timestamp, "Asia/Shanghai");
// converted is "2023-04-13 13:45:00"
var converted2 = convertUTCToTimeZone(timestamp, "UTC+05:30:00");
// converted2 is "2023-04-13 11:15:00"
```

*Converting the time by using the actualBusinessTimestamp of actualEvent and TimeZoneID of a location*

```
var location = getLocationByAltKey(theLocationAltKey);
var converted = convertUTCToTimeZone(actualEvent.actualBusinessTimestamp, location.TimeZoneId);
```



### 5.2.2 Event to Action External Functions

- **sendTrackingRequest**

Function **sendTrackingRequest** defines extra logic or conditions before sending to the visibility provider.

You can use this function to send tracking requests to multiple business partners to report events on the same tracked process. To achieve this, you must add the `receiver` parameter to all tracking requests. Using this parameter only in some requests is not supported.

The function can also log the payload sent to the tracking API when:

- calling the tracking API failed,
- or the [Event to Action Log of the model is enabled](#) in the Manage Models app.

After the message processing, the logs are displayed on the [Correlation Details](#) page -> [Process History](#) tab -> [View Logs](#) button -> [Event to Action External](#) phase in the *Manage Message Logs* app and *Manage Event Logs* app.

```
/**send tracking request to the visibility provider
 * @param isResourceTP true = update the resource tracked process, false = not
 * @optional param trackingRequestConfig a configuration object that provides the visibility
 provider with configuration parameters.
 * @optional param receiver the receiver's identifier (including LBN ID, SCAC code, or
 others). It should follow the pattern "Type#ID", such as SCAC#MAEU or LBN#10002419. If you
 don't add "Type#", the system will identify it as an LBN ID. The value cannot be null.
 */
sendTrackingRequest(isResourceTP, trackingRequestConfig, receiver)
```

For example:

```
sendTrackingRequest(true);
```

The structure of `trackingRequestConfig` should be as follows:

```
{
  "trackingSettings": {
    "trackShipment": [boolean]
    "trackCarbonEmission": [boolean]
  }
}
```

You can choose to track shipments or carbon emissions or both. The default value of `trackShipment` is `true`. The default value of `trackCarbonEmission` is `false`.

For example:

```
var trackingRequestConfig = toObject("{}");
var trackingSettings = toObject("{}");
// enable tracking carbon emission
trackingSettings.trackCarbonEmission = true;
trackingSettings.trackShipment = false;
trackingRequestConfig.trackingSettings = trackingSettings;
var receiver = "10002419";
sendTrackingRequest(true, trackingRequestConfig, receiver);
```

- **forwardCurrentTP**

Function **forwardCurrentTP** forwards the current tracked process, actual event and planed event in JSON format to a destination through HTTP request.

To use this function, you should first configure the destination in the [Destinations](#) section of your subaccount on SAP BTP cockpit. You can define the key-value pair in the [Additional Properties](#) section. Note that properties should be defined as below:

*"sap.lbn.gtt.header."+<key> : <value>*

And <key> : <value> will be added in the header of the forward request.

If the destination is protected by CSRF token, add the following properties in the [Additional Properties](#) section:

Property	Description
<i>sap.lbn.gtt.csrf.header</i>	Mandatory. The header is used to store the value of CSRF token. For the header, you can provide your own based on your server's configuration. In most cases, it should be <i>x-csrf-token</i> .
<i>sap.lbn.gtt.csrf.method</i>	For the method, you can use GET or HEAD. The default is GET if not provided.
<i>sap.lbn.gtt.csrf.url</i>	The URL is used to fetch CSRF token. The default is the main URL configured in the destination if not provided.

For more on the request body structure, refer to sub-section **6.4**.

```
/**forward current tracked process to a destination
 * @param destinationName the destination name
 * @param destination in PaaSAccount or not true = in, false = not
 * @param withOldTPandPE true = forward old tracked process and planned event, false = not
 * @param execute function asynchronously or not true = asynchronously, false = not
 * @returns null
 */
forwardCurrentTP(destinationName,inPaaSAccount,withOldTPandPE)
```

For example:

```
forwardCurrentTP("destinationName",true,false,false);
```

- **forwardCustomPayload**

Function **forwardCustomPayload** forwards customized payload to external systems.

To use this function, you should first configure the destination in the [Destinations](#) section of your subaccount on SAP BTP cockpit. You can select either the basic authentication or client certificate authentication to authenticate messages. You can also define the key-value pair in the [Additional Properties](#) section. Note that properties should be defined as below:

`"sap.lbn.gtt.header."+<key> : <value>`

And `<key> : <value>` will be added in the header of the forward request.

Also note that the hostname URL of the destination shouldn't end with "/" and the parameter should always start with "/".

If the destination is protected by CSRF token, add the following properties in the [Additional Properties](#) section:

Property	Description
<code>sap.lbn.gtt.csrf.header</code>	Mandatory. The header is used to store the value of CSRF token. For the header, you can provide your own based on your server's configuration. In most cases, it should be <code>x-csrf-token</code> .
<code>sap.lbn.gtt.csrf.method</code>	For the method, you can use GET or HEAD. The default is GET if not provided.
<code>sap.lbn.gtt.csrf.url</code>	The URL is used to fetch CSRF token. The default is the main URL configured in the destination if not provided.

```

/**forward customized payload to external systems
 * @param destinationName the destination name
 * @param customPayload contains user-defined key and value and sends to destination service
as JSON file
 * @param path point to the interface
 */
forwardCustomPayload(destinationName, customPayload, path)

```

For example:

```

var destinationName = "TO_QW9_170";
var customPayload = toObject("{}");
customPayload.trackingId = "asdasd";
customPayload.lastReportedEvent = actualEvent;
customPayload.myArray = ["ABC", "DEF"];
var path = "/path";
forwardCustomPayload (destinationName, customPayload, path);

```

- **forwardEventToTM**

Function **forwardEventToTM** forwards one or multiple events back to SAP Transportation Management (SAP TM). You can also choose to forward attachments provided by visibility providers if the mapping is maintained in the [Visibility Provider Integration](#) tab of the MM app.

To use this function, you should first configure the destination in the [Destinations](#) section of your subaccount on SAP BTP cockpit. You can define the key-value pair in the [Additional Properties](#) section. Note that properties should be defined as below:

*"sap.lbn.gtt.header."<key> : <value>*

And *<key> : <value>* will be added in the header of the forward request.

If the destination is protected by CSRF token, add the following properties in the [Additional Properties](#) section:

Property	Description
<i>sap.lbn.gtt.csrf.header</i>	Mandatory. The header is used to store the value of CSRF token. For the header, you can provide your own based on your server's configuration. In most cases, it should be <i>x-csrf-token</i> .
<i>sap.lbn.gtt.csrf.method</i>	For the method, you can use GET or HEAD. The default is GET if not provided.
<i>sap.lbn.gtt.csrf.url</i>	The URL is used to fetch CSRF token. The default is the main URL configured in the destination if not provided.

The function can also log the payload sent to SAP TM when the [Event to Action Log of the model is enabled](#) in the Manage Models app. After the message processing, the logs are displayed on the [Correlation Details](#) page -> [Process History](#) tab -> [View Logs](#) button -> [Event to Action External](#) phase in the *Manage Message Logs* app and *Manage Event Logs* app.

#### Forward a single event:

```
/**forward actual events back to TM system
 * @optional param messageHeader (creationDateTime, recipientBusinessSystemID,
 recipientPartyLBNNNo, senderPartyInternalID, senderPartyStandardID) used for soap message's
 header. For details, see the meessageHeader table below.
 * @optional param event (eventTypeCode, eventReasonCode, eventReasonText,
 referenceEventTypeCode, recipientSystemLocationInternalID, stopId, expectedDateTime,
 estimatedDateTime, actualDateTime, transportationOrderID, forwardAttachments) used for
 building events sent to TM system.
 * @param destinationName the destination name
 */
forwardEventToTM (messageHeader, event, destinationName);
```

#### Forward multiple events:

```
/**forward actual events back to TM system
 * @optional param messageHeader (creationDateTime, recipientBusinessSystemID,
 recipientPartyLBNNNo, senderPartyInternalID, senderPartyStandardID) used for soap message's
 header. For details, see the meessageHeader table below.
 * @param eventArray (event1, event2, ...) The event array contains multiple event objects.
 Each event object uses the parameters as described in the event table below. The eventArray
 can be empty, which means that no events will be sent back to the TM system.
 * @param destinationName the destination name
 */
forwardEventToTM (messageHeader, eventArray, destinationName);
```

*messageHeader table:*

Parameter (Field)	Description	Value	Default Logic
creationDateTime	The point in time when the event is created.	Use function <b>toTimestamp</b> to get the value, such as 2020-11-03T05:47:25.402Z.	Use the current UTC time.
recipientBusinessSystemID			Use the value of the current tracked process' logicalSystem field.
recipientPartyStandardID			Use the value of the current tracked process' partyId field.  (Note: if the value contains the prefix "LBN#", the prefix will be removed.)
senderPartyInternalID			Use the value of the current tracked process' carrierInternal field.
senderPartyStandardID			Use the value of the current tracked process' serviceAgentLbnId field.  (Note: if the value contains the prefix "LBN#", the prefix will be removed.)

*event table:*

Parameter (Field)	Description	Value	Default Logic
eventTypeCode	A coded representation of the type of a transportation event like Departure, Arrival, Proof of Delivery.	For example, in a road freight order, typically one of the following code values can occur:  Arrival: ARRIV_DEST Delayed: DELAYED Departure: DEPARTURE Proof of Delivery: POD Proof of Pickup: POPU Loading Begin: LOAD_BEGIN Loading End: LOAD_END Coupling: COUPLING Decoupling: DECOUPLING	Follow the mapping of Event Type to Event Code in the IDOC Integration tab of the MM app.
eventReasonCode	A coded representation of the reason of a transportation event.	For example, in a road freight order, typically one of the following code values can occur:  Breakdown: BRKDWN Others: OTHERS Traffic: TRAFFC Waiting at Location: WTGLOC	Use the value of the current event's eventReasonCode field.
eventReasonText	A text string that contains the reason of a transportation event.	Free input	Use the value of the current event's eventReasonText field.
referenceEventTypeCode	A coded representation of a specific milestone referring to another event during the transport execution. For example, a	For example, in a road freight order, typically one of the following code values can occur:  Arrival: ARRIV_DEST Departure: DEPARTURE Proof of Delivery: POD	For delayed events and events that inherit GTTDelayedEvent, follow the mapping of refPlannedEventType to Event Code in the IDOC Integration tab of the MM app.

## Guide for Model Administrators

	delayed event refers to an arrival event.	Proof of Pickup: POPU Loading Begin: LOAD_BEGIN Loading End: LOAD_END Coupling: COUPLING Decoupling: DECOUPLING	For other events, this field will be empty.
recipientSystemLocationInternalID	A proprietary identifier for a location where the even was reported.		If the current event matches with a planned event, the value will be the planned event's locationAltKey.  Otherwise, the value will be provided by the current event's locationAltKey.
stopId	A proprietary identifier for a stop where the event was reported.	stopID in your SAP TM system	The field will be empty.
expectedDateTime	The point in time when the event is expected to occur.	Use the function <b>toTimestamp</b> to get the value.	If the current event matches with a planned event, the value will be the planned event's planBusinessTimestamp.  Otherwise, the value will be empty.
estimatedDateTime	The point in time when the event is estimated to occur.	Use the function <b>toTimestamp</b> to get the value.	The field will be empty.
actualDateTime	The point in time when the event occurs.	Use the function <b>toTimestamp</b> to get the value.	Use the value of the current event's actualBusinessTimestamp field.
transportationOrderID	A unique identifier for a transportation order.	The Freight Order Number in your SAP TM system.	Use the current event's tracking ID.
forwardAttachments	When sending a specific event to SAP TM, you can choose to send attachments with the event or not.  Note: If you use the event array as the input, this parameter only takes effect in its own event object.	<b>true:</b> the event with all its attachments will be sent to SAP TM.  <b>false:</b> only the event will be sent to SAP TM.	If you use the event object as the input, the default value is true.  If you use the event array as the input, the default value is false.

### Examples:

#### Forward a single event:

```

if ( "com.tenantname.gtt.app.sofvpauto.Shipment.Arrival" == actualEvent.eventType )
{
    var destinationName = "TO_BF9_960";
    var msgObj = toObject("{}");
    msgObj.creationDateTime = "2020-11-03T05:47:25.402Z";
    msgObj.recipientBusinessSystemID= "BF9_960";
    msgObj.recipientPartyLBNo = "10000786";
    msgObj.senderPartyInternalID = "LBNG4ZC02";
    msgObj.senderPartyStandardID = " 10000088";
}

```

## SAP Business Network Global Track and Trace

```
var eventObj = toObject("{}");
eventObj.eventReasonText = "test event";
eventObj.eventTypeCode = "ARRIV_DEST";
eventObj.recipientSystemLocationInternalID = "LBN-E2E-03";
forwardEventToTM(msgObj, eventObj, destinationName);
}
```

## Guide for Model Administrators

### Forward multiple events:

```
// forwarding estimated timestamp to TM starts
eventForwardToTM = true;
function getPlannedBusinessTime(etaEventType, etaLocationAltKey){
    var i_plannedEvents = 0;
    var z_plannedBusinessTimestamp;
    while (i_plannedEvents < trackedProcess.plannedEvents.size()) {
        if (trackedProcess.plannedEvents[i_plannedEvents].eventType == etaEventType &&
            trackedProcess.plannedEvents[i_plannedEvents].locationAltKey == etaLocationAltKey)
        {
            z_plannedBusinessTimestamp =
trackedProcess.plannedEvents[i_plannedEvents].plannedBusinessTimestamp;
        }
        i_plannedEvents = i_plannedEvents + 1;
    }
    return z_plannedBusinessTimestamp;
}
function generateEventObjects(){
    var etaEventObject = toObject("{}");
    etaEventObject.eventReasonText = "sync back estimated date time from GTT";
    innerEventTypeInfo = getEventTypeInfo(actualEvent.estimatedTimestamps[i].eventType, "en");
    if (innerEventTypeInfo != null && innerEventTypeInfo.idocMapping != null &&
innerEventTypeInfo.idocMapping.erpEventCode != null) {
        etaEventObject.eventTypeCode = "EVENT_EST_UPD";
        innerLocationAltkey =
parseLocationAltKey(actualEvent.estimatedTimestamps[i].locationAltKey);
        etaEventObject.recipientSystemLocationInternalID = innerLocationAltkey.id;
        etaEventObject.estimatedDateTime =
actualEvent.estimatedTimestamps[i].estimatedTimestamp;
        etaEventObject.referenceEventTypeCode = innerEventTypeInfo.idocMapping.erpEventCode;
        etaEventObject.expectedDateTime =
getPlannedBusinessTime(actualEvent.estimatedTimestamps[i].eventType,
actualEvent.estimatedTimestamps[i].locationAltKey);
        eventObjects.push(etaEventObject);
    }
}
if (DELETION_EVENT != actualEvent.eventType && SHIPMENT_TP == trackedProcess.trackedProcessType
&& SHIPMENT_TRACKING_ID_TYPE == actualEvent.trackingIdType && TOR_SHIPMENT_TYPE ==
trackedProcess.shipmentType) {
    if (actualEvent.messageSourceType == "GTT:WS" || actualEvent.messageSourceType == "GTT:VP")
    {
        if (trackedProcess != null && eventForwardToTM &&
actualEvent.containsProperty("estimatedTimestamps")) {
            var i = 0;
            var innerEventTypeInfo;
            var innerLocationAltkey;
            var eventObjects = [];
            while (i < actualEvent.estimatedTimestamps.size()) {
                generateEventObjects();
                i = i + 1;
            }
            forwardEventToTM(msgObj, eventObjects, destinationTMName);
        }
    }
}
}
//forwarding estimated timestamp to TM ends
```



- **updateInboundDeliveryInERP**

Function **updateInboundDeliveryInERP** calls ERP SOAP interfaces to update the inbound delivery in SAP ERP. Currently only the field *Planned Delivery Date* is supported for updates. For more, see the “Synchronize ETA of the Inbound Delivery Back to ERP System” chapter of the guide *How to Send Documents from SAP Business Network Global Track and Trace to Other Systems*.

But if your SAP ERP system is already integrated with SAP Extended Warehouse Management (SAP EWM), this function will not operate.

To use this function, you should first configure the destination in the [Destinations](#) section of your subaccount on SAP BTP cockpit. You can define the key-value pair in the [Additional Properties](#) section. Note that properties should be defined as below:

*“sap.lbn.gtt.header.”+<key> : <value>*

And <key> : <value> will be added in the header of the forward request.

If the destination is protected by CSRF token, add the following properties in the [Additional Properties](#) section:

Property	Description
<i>sap.lbn.gtt.csrf.header</i>	Mandatory. The header is used to store the value of CSRF token. For the header, you can provide your own based on your server's configuration. In most cases, it should be <i>x-csrf-token</i> .
<i>sap.lbn.gtt.csrf.method</i>	For the method, you can use GET or HEAD. The default is GET if not provided.
<i>sap.lbn.gtt.csrf.url</i>	The URL is used to fetch CSRF token. The default is the main URL configured in the destination if not provided.

```

/**update Inbound Delivery in ERP system
 * @optional param messageHeader (creationDateTime, recipientBusinessSystemID,
recipientPartyLBNo, senderPartyInternalID, senderPartyStandardID) used for soap message's
header
 * @optional param event (trackingId, deliveryVersion, plannedDeliveryTime) used for
building events sent to ERP system to update related inbound delivery
 * @param destinationUpdateInboundDeliveryInERP the destination name
 */
updateInboundDeliveryInERP (messageHeader, event, destinationUpdateInboundDeliveryInERP);

```

*messageHeader table:*

Parameter (Field)	Description	Value	Sample	Default Logic
creationDateTime	The point in time when the event is created.	Use function <b>toTimestamp</b> to get the value.	2020-11-03T05:47:25.402Z	Use the current UTC time.
recipientBusinessSystemID			QM7_910	
recipientPartyStandardID			10000042	
senderPartyInternalID			LBNE2EC02	
senderPartyStandardID			10000044	

## Guide for Model Administrators

event table:

Parameter (Field)	Description	Value	Sample	Default Logic
trackingId	An ID representation of the delivery number in ERP system	An existing inbound delivery number in ERP system, refer to field VBELN in table LIKP in ERP system		
deliveryVersion	A number representation of the current version of inbound delivery	The delivery version formation for the specify inbound delivery, refer to field DLV_VERSION in table LIKP in ERP system	0001	
plannedDeliveryTime	The new planned delivery date to be updated	Use function <b>toTimestamp</b> to get the value.	2021-05-15T08:00:00.213Z	

For example:

```
if ( "com.tenantname.gtt.app.sofvpauto.InboundDelivery.InboundDeliveryEvent" ==  
actualEvent.eventType )  
{  
    var destinationName = " TO_QW9_170_UPDATE_INBDLV";  
    var msgObj = toObject("{}");  
    msgObj.creationDateTime = "2020-11-03T05:47:25.402Z";  
    msgObj.recipientBusinessSystemID= "QW9_170";  
    msgObj.recipientPartyLBNo = "10000786";  
    msgObj.senderPartyInternalID = "LBNG4ZC02";  
    msgObj.senderPartyStandardID = " 10000088";  
    var eventObj = toObject("{}");  
    eventObj.trackingId = "187019198";  
    eventObj.deliveryVersion = "0001";  
    eventObj.plannedDeliveryTime = "2021-05-15T08:00:00.213Z";  
    updateInboundDeliveryInERP(msgObj, eventObj, destinationName);  
}
```

## Object Structure References

## 6.1 Example of trackedProcess Object

```

{
  "isCanceled": false,
  "freightOrderId": "test_61",
  "cargoWeightUOM": "KG",
  "processStatus_code": "EARLY",
  "references": [],
  "senderPartyId": null,
  "plannedArrivalDateTime": "2020-01-11T08:00:00.213Z",
  "departureLocationId":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2010",
  "shipperLbnId": "10000034",
  "id": "58978ad7-20b4-11ea-a0f2-51352b993cd3",
  "trafficDirection": "NOR",
  "numberOfUnloadingStops": 2,
  "eventReasonText": null,
  "partyId": "LBN#10000034",
  "integratedWithVP": null,
  "trackingId": "test_61",
  "longitude": null,
  "creationDateTime": "2019-12-17T10:02:34.075Z",
  "goodsValueCurr": "EUR",
  "messageSourceType": null,
  "actualBusinessTimeZone": null,
  "cargoQuantity": 5.000000000000000,
  "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.FreightOrderEvent",
  "plannedArrivalDateTimeZone": "CET",
  "cloneInstanceId": "a6eeb2a8-3f82-456b-8863-5149572e604a",
  "cargoWeight": 70.00000000000000,
  "incoterm": "EXW",
  "nextStopId": null,
  "orderingPartyUserId": "10000034",
  "stops": [
    {
      "locationTimeZone": "UTC+3",
      "freightOrderEvent_id": "58978ad7-20b4-11ea-a0f2-51352b993cd3",
      "plannedDepartureAt": "2020-01-06T08:00:00.213Z",
      "plannedArrivalAt": "2020-01-06T08:00:00.213Z",
      "departureDateTimeDSTIndicator": null,
      "arrivalDateTimeDSTIndicator": null,
      "stopId": 30,
      "sequentialNumber": 3,
      "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011"
    }
  ],
  "locationAltKey": null,
  "plannedDepartureDateTimeZone": "CET",
  "meansOfTransport": "Truck",
  "carrierExternalId": "LBN_CAR001",
  "departureStopId": 1,
  "scheme": "xri://sap.com/id",
  "logicalSystem": "Q8JCLNT774",
  "carrierLbnId": "10000012",
  "cargoVolumeUOM": "m3",
  "createdByUser": "gtt_v2_dev@sap.com",
  "latitude": null,
  "shippingType": "FTL",
  "goodsValue": 90.000000,
  "numberOfLoadingStops": 2,
  "cargoVolume": 12.000000000000000,
  "plannedTotalDistance": 200.000000,
  "registrationCountry": "DE",
  "vehicle": "0023133",
  "plannedDepartureDate": "2019-12-30",
  "plannedTotalDistanceUOM": null,

```

## Guide for Model Administrators

```
"plannedNetDuration": "PT1H",
"eventReasonCode": null,
"altKey": "xri://sap.com/id:LBN#10000034:Q8JCLNT774:FreightOrder:test_61",
"originalDocuments": [
  {
    "number": "1234",
    "freightOrderEvent_id": "58978ad7-20b4-11ea-a0f2-51352b993cd3"
  }
],
"plannedArrivalDate": "2019-12-06",
"estimatedArrivalTime": "2020-01-06T08:00:00.213Z",
"shipperExternalId": "LBN_SHIPP1",
"plannedGrossDuration": "PT1H",
"numberOfVisits": 5,
"businessReferenceNo": [
  {
    "itemId": "item123",
    "freightOrderEvent_id": "58978ad7-20b4-11ea-a0f2-51352b993cd3",
    "documentTypeCode": "pdf",
    "itemTypeCode": "txt",
    "documentId": "refer1234"
  }
],
"subaccountId": "86cbb49a-0deb-4c74-9c85-9a85908569f1",
"actualTechnicalTimestamp": "2019-12-17T10:02:34.075Z",
"dangerousGoods": "1",
"plannedEvents": [
  {
    "isFinalPlannedEvent": false,
    "plannedBizTsEarliest": "2020-01-05T07:50:00.213Z",
    "plannedBusinessTimestamp": "2020-01-05T08:00:00.213Z",
    "latitude": null,
    "plannedBusinessTimeZone": "CET",
    "plannedTechnicalTimestamp": "2020-01-05T11:00:00.213Z",
    "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
    "event_id": "58978ad7-20b4-11ea-a0f2-51352b993cd3",
    "id": "58978ad8-20b4-11ea-a0f2-85331debb9ae",
    "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011",
    "plannedTechTsLatest": "2020-01-05T11:10:00.213Z",
    "plannedTechTsEarliest": "2020-01-05T10:50:00.213Z",
    "plannedBizTsLatest": "2020-01-05T08:10:00.213Z",
    "eventMatchKey": "30",
    "longitude": null
  }
],
"cargoQuantityUOM": "PAL",
"arrivalLocationId":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2001",
"plannedDepartureDateTime": "2019-12-30T08:00:00.213Z",
"actualBusinessTimestamp": "2019-12-29T08:00:00.213Z",
"estimatedArrivalTimeZone": null,
"registrationNumber": "HDG123",
"trackingIdType": "FreightOrder",
"arrivalStopId": 40,
"eventMatchKey": null
}
```

**6.2 Example of actualEvent Object**

```

{
  "scheme": "xri://sap.com/id",
  "logicalSystem": "Q8JCLNT774",
  "references": [],
  "createdByUser": "gtt_v2_dev@sap.com",
  "latitude": 50.123600000,
  "senderPartyId": null,
  "id": "bf1a5455-20b3-11ea-a0f2-c95cd3b8a2f4",
  "eventReasonCode": null,
  "altKey": "xri://sap.com/id:LBN#10000034:Q8JCLNT774:FreightOrder:test_676",
  "eventReasonText": "test_676ArrivalEvent_INTERMEDIATE",
  "partyId": "LBN#10000034",
  "estimatedArrivalTime": null,
  "trackingId": "test_676",
  "longitude": 100.123000000,
  "creationDateTime": "2019-12-17T09:58:16.563Z",
  "subaccountId": "86cbb49a-0deb-4c74-9c85-9a85908569f1",
  "messageSourceType": null,
  "actualTechnicalTimestamp": "2019-12-17T09:58:16.563Z",
  "actualBusinessTimeZone": "UTC+8",
  "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
  "cloneInstanceId": "a6eeb2a8-3f82-456b-8863-5149572e604a",
  "actualBusinessTimestamp": "2019-12-12T10:00:00.213Z",
  "estimatedArrivalTimeZone": null,
  "trackingIdType": "FreightOrder",
  "nextStopId": null,
  "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011",
  "eventMatchKey": "30"
}

```

### 6.3 Example of plannedEvent Object

```
{
  "isFinalPlannedEvent": false,
  "lastProcessEventDirectory_id": "bf335a96-20b3-11ea-a0f2-4133d7ffdbfd",
  "process_id": "f0981857-f825-5a24-a664-f3b57fda7ae5",
  "nextOverdueDetection": "2020-01-05T11:10:00.213Z",
  "payloadSequence": 4,
  "plannedBizTsEarliest": "2020-01-05T07:50:00.213Z",
  "plannedBusinessTimestamp": "2020-01-05T08:00:00.213Z",
  "latitude": null,
  "plannedBusinessTimeZone": "CET",
  "plannedTechnicalTimestamp": "2020-01-05T11:00:00.213Z",
  "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
  "eventStatus_code": "EARLY_REPORTED",
  "overdueDetectionCounter": 0,
  "id": "32468a72-20b3-11ea-a0f2-3f0c43e0c83f",
  "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011",
  "plannedTechTsLatest": "2020-01-05T11:10:00.213Z",
  "plannedTechTsEarliest": "2020-01-05T10:50:00.213Z",
  "plannedBizTsLatest": "2020-01-05T08:10:00.213Z",
  "eventMatchKey": "30",
  "longitude": null
}
```

**6.4 Example of forwardCurrentTP request body**

```

{
  "trackedProcess": {
    "isCanceled": false,
    "freightOrderId": "test_175",
    "cargoWeightUOM": "KG",
    "processStatus_code": "EARLY",
    "plannedArrivalDateTime": "2020-01-11T08:00:00.213Z",
    "departureLocationId":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2010",
    "shipperLbnId": "10000034",
    "processEvents": [
      {
        "process_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "event_id": "e7cdc7b2-20ba-11ea-9b6b-9d8006c8f832",
        "plannedEvent_id": null,
        "correlationType": "UNPLANNED_PROCESS_CREATION_UPDATE",
        "id": "e800be97-20ba-11ea-9b6b-9532102f6fe0"
      },
      {
        "process_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "event_id": "42ebe418-20bb-11ea-9b6b-73d289befbc1",
        "plannedEvent_id": "e7ec9a55-20ba-11ea-9b6b-0b9d996dd349",
        "correlationType": "EARLY_REPORTED",
        "id": "42fdb6e9-20bb-11ea-9b6b-b9f28029c9c2"
      }
    ],
    "id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
    "trafficDirection": "NOR",
    "numberOfUnloadingStops": 2,
    "trackedProcessType": "com.sap.gtt.app.tfo.FreightOrderModel.FreightOrderProcess",
    "partyId": "LBN#10000034",
    "integratedWithVP": null,
    "longitude": 100.123000000,
    "trackingId": "test_175",
    "creationDateTime": "2019-12-17T10:49:31.527Z",
    "goodsValueCurr": "EUR",
    "executionStatus": "INTERMEDIATE",
    "lastBusinessDateTime": "2019-12-12T10:00:00.213Z",
    "cargoQuantity": 5.00000000000000,
    "plannedArrivalDateTimeZone": "CET",
    "cloneInstanceId": "a6eeb2a8-3f82-456b-8863-5149572e604a",
    "version": 3,
    "cargoWeight": 70.00000000000000,
    "incoterm": "EXW",
    "nextStopId": null,
    "orderingPartyUserId": "10000034",
    "stops": [
      {
        "locationTimeZone": "UTC+1",
        "plannedDepartureAt": "2019-12-28T08:00:00.213Z",
        "plannedArrivalAt": "2019-12-28T08:00:00.213Z",
        "departureDateTimeDSTIndicator": null,
        "arrivalDateTimeDSTIndicator": null,
        "stopId": 10,
        "freightOrderProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "sequentialNumber": 1,
        "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2001"
      },
      {
        "locationTimeZone": "UTC+2",
        "plannedDepartureAt": "2019-12-30T08:00:00.213Z",
        "plannedArrivalAt": "2019-12-30T08:00:00.213Z",
        "departureDateTimeDSTIndicator": null,
        "arrivalDateTimeDSTIndicator": null,
        "stopId": 20,
        "freightOrderProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "sequentialNumber": 2,

```

## Guide for Model Administrators

```
        "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2010"
    },
    {
        "locationTimeZone": "UTC+3",
        "plannedDepartureAt": "2020-01-06T08:00:00.213Z",
        "plannedArrivalAt": "2020-01-06T08:00:00.213Z",
        "departureDateTimeDSTIndicator": null,
        "arrivalDateTimeDSTIndicator": null,
        "stopId": 30,
        "freightOrderProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "sequentialNumber": 3,
        "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011"
    },
    {
        "locationTimeZone": "UTC+4",
        "plannedDepartureAt": "2020-01-10T08:00:00.213Z",
        "plannedArrivalAt": "2020-01-10T08:00:00.213Z",
        "departureDateTimeDSTIndicator": null,
        "arrivalDateTimeDSTIndicator": null,
        "stopId": 40,
        "freightOrderProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "sequentialNumber": 9,
        "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2001"
    }
],
"plannedDepartureDateTimeZone": "CET",
"lifeCycleStatus": "BUSINESS_ACTIVE",
"meansOfTransport": "Truck",
"carrierExternalId": "LBN_CAR001",
"departureStopId": 1,
"scheme": "xri://sap.com/id",
"logicalSystem": "Q8JCLNT774",
"carrierLbnId": "10000012",
"cargoVolumeUOM": "m3",
"createdByUser": "gtt_v2_dev@sap.com",
"latitude": 50.123600000,
"shippingType": "FTL",
"goodsValue": 90.000000,
"lastChangedByUser": "gtt_v2_dev@sap.com",
"numberOfLoadingStops": 2,
"cargoVolume": 12.000000000000000,
"plannedTotalDistance": 200.000000,
"registrationCountry": "DE",
"vehicle": "0023133",
"plannedDepartureDate": "2019-12-30",
"plannedTotalDistanceUOM": null,
"plannedNetDuration": "PT1H",
"lastChangeDateTime": "2019-12-17T10:52:04.312Z",
"altKey": "xri://sap.com/id:LBN#10000034:Q8JCLNT774:FreightOrder:test_175",
"originalDocuments": [
    {
        "number": "1234",
        "freightOrderProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb"
    }
],
"plannedArrivalDate": "2019-12-06",
"estimatedArrivalTime": null,
"shipperExternalId": "LBN_SHIPP1",
"plannedGrossDuration": "PT1H",
"numberOfVisits": 5,
"businessReferenceNo": [
    {
        "itemId": "item123",
        "documentTypeCode": "pdf",
        "itemTypeCode": "txt",
        "freightOrderProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
```



## SAP Business Network Global Track and Trace

```

        "documentId": "refer1234"
    }
],
"subaccountId": "86cbb49a-0deb-4c74-9c85-9a85908569f1",
"dangerousGoods": "1",
"plannedEvents": [
    {
        "isFinalPlannedEvent": null,
        "lastProcessEventDirectory_id": null,
        "process_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "nextOverdueDetection": "2020-01-06T11:10:00.213Z",
        "payloadSequence": 2,
        "plannedBizTsEarliest": "2020-01-06T07:50:00.213Z",
        "plannedBusinessTimestamp": "2020-01-06T08:00:00.213Z",
        "latitude": null,
        "plannedBusinessTimeZone": "CET",
        "plannedTechnicalTimestamp": "2020-01-06T11:00:00.213Z",
        "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
        "eventStatus_code": "PLANNED",
        "overdueDetectionCounter": 0,
        "id": "e7f043d6-20ba-11ea-9b6b-c3ad279042b4",
        "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2001",
        "plannedTechTsLatest": "2020-01-06T11:10:00.213Z",
        "plannedTechTsEarliest": "2020-01-06T10:50:00.213Z",
        "plannedBizTsLatest": "2020-01-06T08:10:00.213Z",
        "eventMatchKey": "40",
        "longitude": null
    },
    {
        "isFinalPlannedEvent": false,
        "lastProcessEventDirectory_id": "42fdbe69-20bb-11ea-9b6b-b9f28029c9c2",
        "process_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "nextOverdueDetection": "2020-01-05T11:10:00.213Z",
        "payloadSequence": 1,
        "plannedBizTsEarliest": "2020-01-05T07:50:00.213Z",
        "plannedBusinessTimestamp": "2020-01-05T08:00:00.213Z",
        "latitude": null,
        "plannedBusinessTimeZone": "CET",
        "plannedTechnicalTimestamp": "2020-01-05T11:00:00.213Z",
        "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
        "eventStatus_code": "EARLY_REPORTED",
        "overdueDetectionCounter": 0,
        "id": "e7ec9a55-20ba-11ea-9b6b-0b9d996dd349",
        "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011",
        "plannedTechTsLatest": "2020-01-05T11:10:00.213Z",
        "plannedTechTsEarliest": "2020-01-05T10:50:00.213Z",
        "plannedBizTsLatest": "2020-01-05T08:10:00.213Z",
        "eventMatchKey": "30",
        "longitude": null
    }
],
"cargoQuantityUOM": "PAL",
"arrivalLocationId":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2001",
"plannedDepartureDateTime": "2019-12-30T08:00:00.213Z",
"estimatedArrivalTimeZone": null,
"registrationNumber": "HDG123",
"trackingIdType": "FreightOrder",
"arrivalStopId": 40,
"trackingIds": [
    {
        "process_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "observedProcess_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
        "validFrom": "1970-01-01T00:00:00Z",
        "validTo": "9999-12-31T23:59:59.999Z"
    }
]
],

```

## Guide for Model Administrators

```
    "lastExecutionStatusChangedAt": "2019-12-12T02:00:00.213Z"
  },
  "actualEvent": {
    "scheme": "xri://sap.com/id",
    "logicalSystem": "Q8JCLNT774",
    "references": [],
    "createdByUser": "gtt_v2_dev@sap.com",
    "latitude": 50.123600000,
    "senderPartyId": null,
    "id": "42ebe418-20bb-11ea-9b6b-73d289befbc1",
    "eventReasonCode": null,
    "altKey": "xri://sap.com/id:LBN#10000034:Q8JCLNT774:FreightOrder:test_175",
    "eventReasonText": "test_175ArrivalEvent_INTERMEDIATE",
    "partyId": "LBN#10000034",
    "estimatedArrivalTime": null,
    "trackingId": "test_175",
    "longitude": 100.123000000,
    "creationDateTime": "2019-12-17T10:52:04.195Z",
    "subaccountId": "86cbb49a-0deb-4c74-9c85-9a85908569f1",
    "messageSourceType": null,
    "actualTechnicalTimestamp": "2019-12-17T10:52:04.195Z",
    "actualBusinessTimeZone": "UTC+8",
    "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
    "cloneInstanceId": "a6eeb2a8-3f82-456b-8863-5149572e604a",
    "actualBusinessTimestamp": "2019-12-12T10:00:00.213Z",
    "estimatedArrivalTimeZone": null,
    "trackingIdType": "FreightOrder",
    "nextStopId": null,
    "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011",
    "eventMatchKey": "30"
  },
  "plannedEvent": {
    "isFinalPlannedEvent": false,
    "lastProcessEventDirectory_id": "42fdb6e69-20bb-11ea-9b6b-b9f28029c9c2",
    "process_id": "7f299cc2-6e04-5ba8-b29d-6b3a53f4c5fb",
    "nextOverdueDetection": "2020-01-05T11:10:00.213Z",
    "payloadSequence": 1,
    "plannedBizTsEarliest": "2020-01-05T07:50:00.213Z",
    "plannedBusinessTimestamp": "2020-01-05T08:00:00.213Z",
    "latitude": null,
    "plannedBusinessTimeZone": "CET",
    "plannedTechnicalTimestamp": "2020-01-05T11:00:00.213Z",
    "eventType": "com.sap.gtt.app.tfo.FreightOrderModel.ArrivalEvent",
    "eventStatus_code": "EARLY_REPORTED",
    "overdueDetectionCounter": 0,
    "id": "e7ec9a55-20ba-11ea-9b6b-0b9d996dd349",
    "locationAltKey":
"xri://sap.com/id:LBN#10000034:Q8JCLNT774:Location:shippingPoint:2011",
    "plannedTechTsLatest": "2020-01-05T11:10:00.213Z",
    "plannedTechTsEarliest": "2020-01-05T10:50:00.213Z",
    "plannedBizTsLatest": "2020-01-05T08:10:00.213Z",
    "eventMatchKey": "30",
    "longitude": null
  }
}
```

**www.sap.com/contactsap**

© 2020 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.