Event Connector

Planon Software Suite

Version: L104

Planon
Building Connections

About this Document

Intended Audience

This document is intended for *Planon Software Suite* users.

Contacting us

If you have any comments or questions regarding this document, please send them to: support@planonsoftware.com.

Document Conventions

Bold
Names of menus, options, tabs, fields and buttons are displayed in bold type.

Italic text
Application names are displayed in italics.

CAPITALS
Names of keys are displayed in upper case.

Special symbols

	Text preceded by this symbol references additional information or a tip.
	Text preceded by this symbol is intended to alert users about consequences if they carry out a particular action in Planon.

Table of Contents

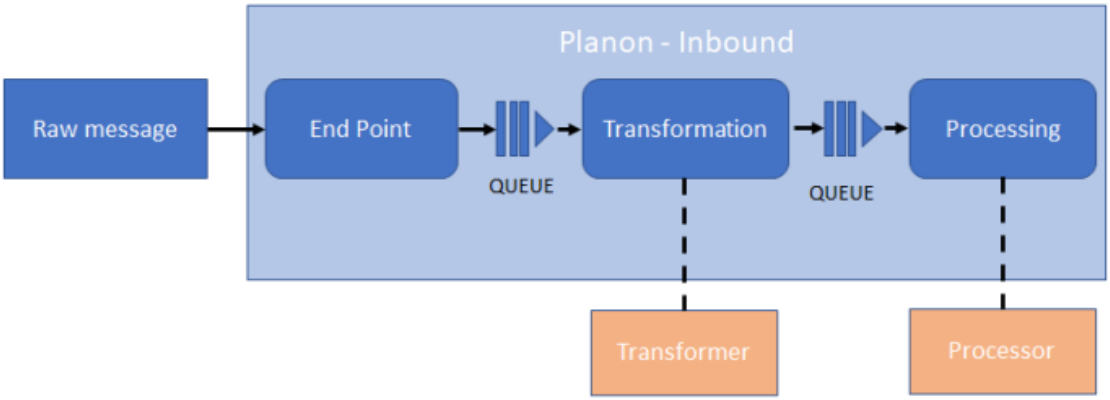
Event connector

Event connector is a gateway between third-party applications and Planon.

It is designed to process *messages* that are sent by another application and which need to be processed in the Planon application, and vice versa. Consequently, there is an inbound stream and an outbound stream.

Inbound

A message can be anything in any format. Consequently, the Planon application needs to be instructed on how to specifically handle messages.



Typically, inbound messages need to be *received*, *transformed* and *processed*. Accordingly, Event connector consists of the following three TSIs (and business objects):

- Inbound raw messages
- Inbound messages
- Business events

Event connector can generically be applied to process all kinds of messages (data), that can subsequently be processed by the Planon application.

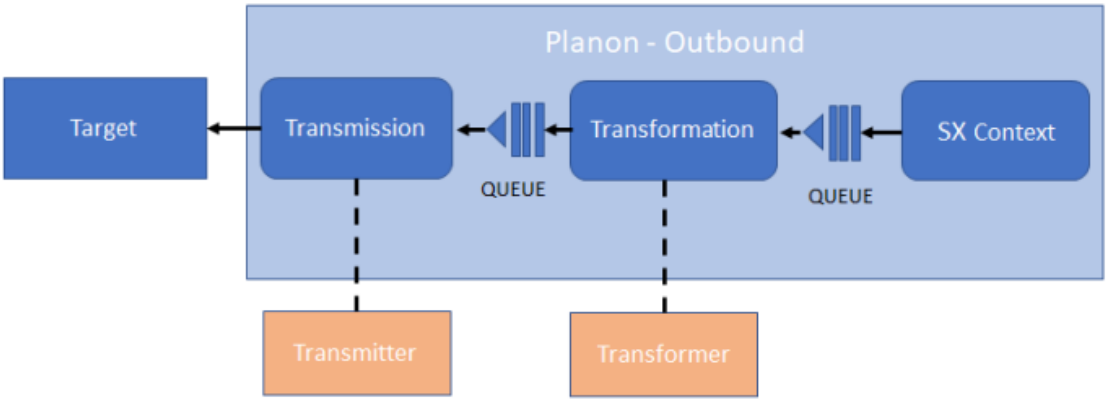
Outbound

Whenever specific message is received, it will be queued for transformation.

Once transformed into an *outbound message*, it will be queued again for transmission. The result will be an *outbound raw message*.

Accordingly, Event connector consists of the following two TSIs (and business objects):

- Outbound messages
- Outbound raw messages



The outbound raw message will subsequently be sent to the target, which can be anything, such as: another application, email, log, REST API, another Planon endpoint...

Event connector

Requirements

When using the Event connector functionality, the following requirements apply:

- User group / user account

  When a message is put on the outbound queue, this is done by using the account that triggers the event creating the outbound message. To process the outbound message, this user needs to be linked to a user group that is linked to the **Event connector** product definition. To prevent that every user (user group) has to be linked to the **Event connector** product definition, an **Event connector admin** system account/user group is used instead.

User group	
Name	Eventconnectoradmin
Function profile	<empty>
Linked product definition	Event connector
User account	
User name	EVENTCONNECTORADMIN
Description	Event connector admin
Language	US English

User group	
User group	Eventconnectoradmin

Concepts

The following topics describe the concepts that are key to understanding the functionality.

Business event

When an inbound message is processed, a *business event* is created.

In contrast to the inbound messages, a business event is directly related to Planon business objects (properties, assets, etc.). Based on the message type, business events can be processed, for example, properties/assets can be created/updated or diagnostics can be carried out that may result in order.

It is not possible to register extensions on a business event as this could corrupt the decision model. When using apps that have this, the extensions are ignored.

Business event fields

Decision model

A decision model is container component for which you can create decision rules.

A decision rule determines if and what action needs to be taken when specific filter criteria are met.

For example:

- Should a property created/updated?
- Should asset created/updated?
- Should qualified diagnostics result in a work order?

When a business event is created and a decision model is linked, all decision rules in the model are executed in the specified sequence.

If a decision rule cannot be assessed, its status changes to *Awaiting assessment*. Upon manual assessment, the status is changed to *Approved* or *Rejected* and the decision model will automatically proceed and take appropriate action.

Use case

Suppose application data is sent to Planon for further assessment. The data (inbound raw messages) contains asset *measurements* that may need to be followed up.

A decision model is set up in the Planon application containing rules for assessing whether action is required. (Avoidable costs >= 2500, Maintenance priority >= 6, Energy priority >= 1 AND Avoidable cost >= 1000, …).

The business events are received are passed through the decision model. If the conditions are met, orders can be created in the Planon application and the result is sent back to the application.

Decision models/rules - fields

Endpoint

An endpoint is the location through which APIs can access and send requests to the Planon application.

For endpoints it is mandatory to have *system* as system type.

Message

The message is created after transforming the raw message.

Consequently, a message represents a transformed version of the raw message. The message is made up of a valid format that can be processed by the Planon application.

The *transformers* for converting a raw message into a message are specific to the interface and need to be specified in an app.

Messages cannot have *system* as system type.

Raw message

The raw message is the actual message that is sent between two applications.

The communication layer must be able to handle the message without having any knowledge of the content of the message (body). The information on how to handle this message is stored in the envelope fields.

The raw message is in JSON format:

```
{
```

```
    "version": "1.0",
    "messageID": "ecfdea0f-da94-4a7f-ac87-0efb80f2575c",
    "from": "Acme Corporation",
    "systemType": "Acme Corporation",
    "messageType": "Diagnostics",
    "messageBody": Any JSON value
}
```

The order of the values in the JSON structure is not mandatory.

Raw messages cannot have *system* as system type.

(Raw) message fields

Transmission strategy

When outbox raw messages are queued, they are waiting to be sent for transmission.

When processing messages, there are two transmission strategies:

- PUSH

  This method simply means that when a message is received, it is automatically sent.

- PULL

  This method means that when a message is received, it will remain queued until it is retrieved (pulled) by the endpoint.

  In order to use the Pull strategy, the EVENTCONNECTORADMIN account needs to be specified, which is the account that will pull the message record out of Planon.

  When an outbound message is created with an account linked, then the strategy defaults to *PULL* (unless *PUSH* is specified).

  For PULL messages, the endpoint can use the following HTTP methods:

  - GET: finds, reads the message and changes the BO status to *Sent*.
  - DELETE: deletes the message from the outbox.

Working with...

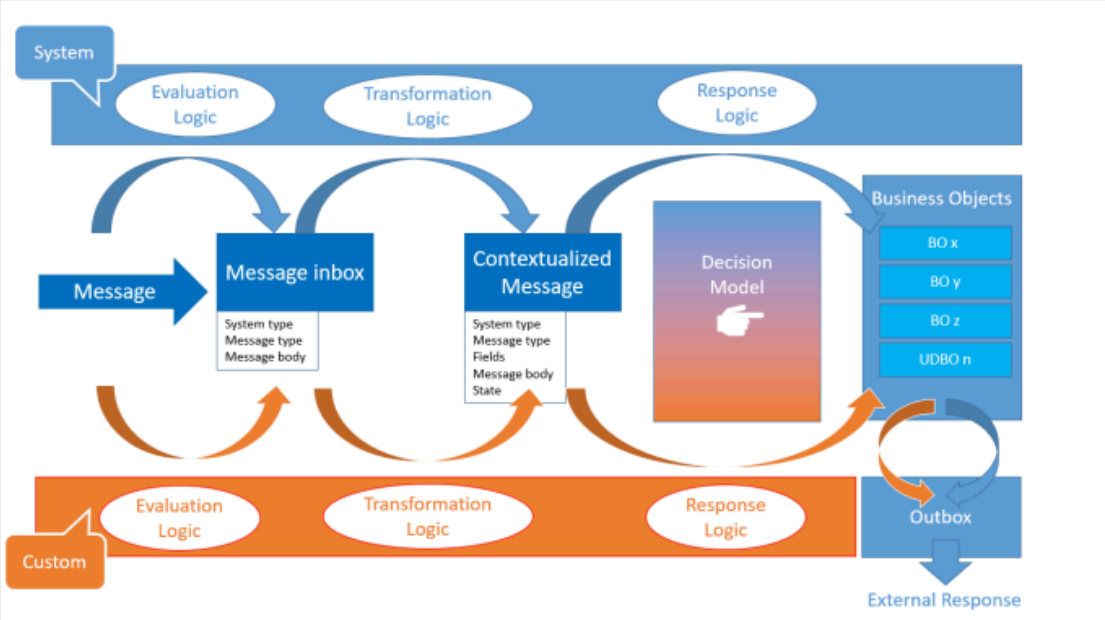
This section describes the various functions available.

Processing messages

Planon's **Event connector** is a generic component that is designed to receive and send messages.

Messages can be anything and they can be received in any format. Therefore, Event Connector will need to process and transform it to make it compatible with Planon.

When further zooming in on processing messages, the following actions take place:



1. *Inbound raw messages* are received from a third-party application.

   By using a system app (or, optionally, a custom app), the data is transformed into an *Inbound message*.
2. The message is further processed and *categorized* into business events and stored in the corresponding user-defined business object (UDBO). Again, this is done by a system app (or, optionally, a custom app).

Messages can be processed by using either a system app or a custom app. By creating and using a ***custom app***, you exert even more control over the data configuration.

Enabling performance monitoring

It is possible to log the performance monitoring data for Event Connector.

This data allows you to analyze the performance of processes involved in handling the sequence of data processing by Event Connector.

To enable monitoring, go to System info > Performance monitoring and set the **Is activated** to **Yes**.

When enabled, a section displaying the Connector monitoring data is included in the log.

In addition to logging message processing information (number of hits, time it takes, etc), the log also specifies the throughput time of the message queue itself:

QueuePerformanceData.TimeElapsedBetweenMessageAdditionAndCompletion

Entry	min(ms)	avg(ms)	max(ms)	#hits	total(ms)
ExternalEventsDLQ	5	9	93	200	1938
InboundMessageQueue.1	46	8276	20081	250	2069114
OutboundMessageQueue.1	39	8262	19630	250	2065698
RawInboundMessageQueue.1	26	6393	19816	250	1598346
RawOutboundMessageQueue.1	59	8255	20280	250	2063907

For an example of the log, see Viewing performance monitoring data in **System Settings**.

Processing messages in queues

The various messages are sequentually processed for further integration into and processing by the Planon application.

To ensure that the communication to the client initiating these messages happens quickly, they are queued in between the processing stages.

Once a message is on a queue, this is acknowledged to the client, which means the client does not have to wait until it is processed. Queuing ensures that messages will not get lost and will be processed in the right order.

Before picking up items from the queue, the respective component finds the corresponding transformer or processor and carries that out.

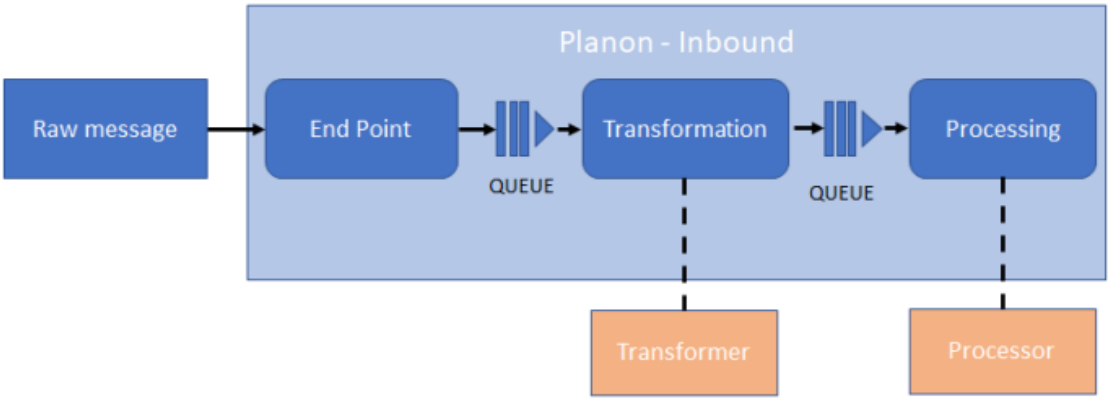
The queueing mechanism ensures that the processing takes place correctly. Each step is either successful (receiving, transformation, processing) and is committed to a subsequent queue and database, or fails and is put on the queue again.

Persistent failures will be put on the queue and retried 9 times before finally being logged as failed.

When the transformation, processing or transmission fails, the **Comment** field of the respective **Inbound/Outbound raw messages** or **Inbound/outbound messages** business object is updated with an error message.

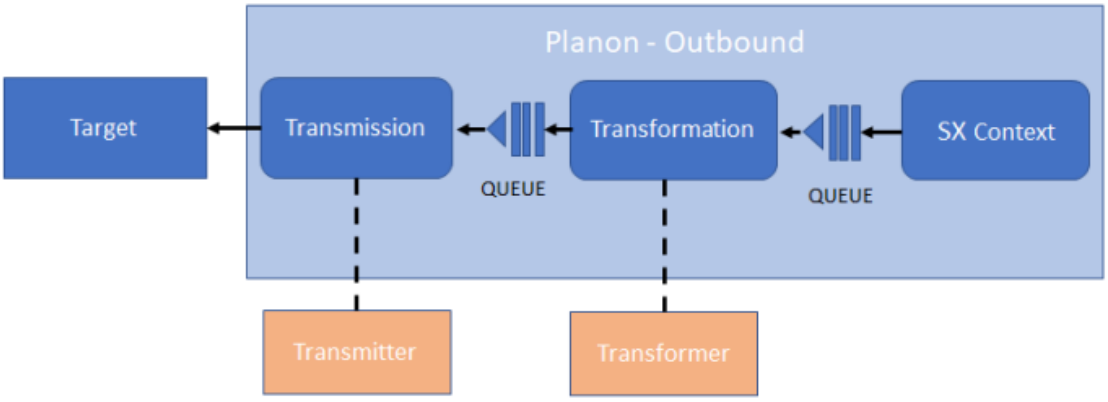
The error will be logged in the business object preceding the step in which the error occurred. For example, if inbound processing fails, the error is logged in the transformation step.

Inbound



- When a raw message is received, it is put on the *Inbound raw message queue* for transformation.
- After transformation, it is put on the *Inbound message queue* for processing.

Outbound



- When a specific SX context is detected, an outbound message is put on the *outbound message queue* for transformation.
- After transformation, the outbound raw message is put on the *outbound raw message queue* for transmission.

Queue is full

When the queue is full, if possible, the user will receive a prompt. In any case, **Queue statistics** will be added to the server log (Wildfly). This will help analyze and troubleshoot issues if they occur:

```
Logging queue statistics information for queue: 'jms.queue.RawInboundMessageQueue.1'

Consumer Count (The number of consumers consuming messages from this queue) = 15

Delivering Count (The number of messages that this queue is currently delivering to its
 consumers) = 19437

Message Count (The number of messages currently in this queue) = 19437

Scheduled Count (The number of scheduled messages in this queue) = 0

Messages Added (The number of messages added to this queue since it was created) = 19437

Server uptime (The amount of time this server/JVM is up and running) = 1 hr

Message inflow (The average message inflow speed for this queue since it was created) =
 19437 msg/hr
```

Security

When dealing with communication and data exchange between applications, security is an important component to consider.

Because of this, the Planon application only accepts an HTTP post containing an *access key* in the header:

```
‘Authorization: PLANONKEY accesskey=<key>’
```

In addition, there is a separate product definition for Event Connector. This makes it possible to create an account that can only access Event Connector and no other part of the Planon application.

For more information about (configuring) access keys, see the Planon WebHelp.

Transformer

A transformer is a software component that is designed for *transforming* messages (inbound or outbound).

These attributes can only contain values with simple types (double, int, string). The transformed attributes can be viewed in the **Inbound/Outbound messages** TSI .

Transformers are specific to the kind of message and consequently need to be custom made by creating an IPnInboundMessageTransformer or IPnOutboundMessageTransformer app and registering it for a systemType and messageType combination.

- If there are multiple transformers for the same combination, they will be picked randomly.
- Transformers are constructed in the *Planon Platform IDE*.

Processor

A processor is a software component that is responsible for modifying/adding business objects in the Planon application.

It reads the transformed messages and, for example, creates a business object.

Processors are specific to the kind of message and consequently need to be custom made by creating an IPnInboundMessageProcessor app and registering it for a systemType and messageType combination.

- If there are multiple processors, all will be executed.
- Processors are written in the *Planon as a Platform IDE*.

Transmitter

A transmisster is a software component that is responsible for sending outbound raw messages.

It (IPnOutboundMessageTransmitter) reads the transformed outbound raw messages from the queue and sends these to the required destination.

Transmitters are constructed in the *Planon Platform IDE*.

Message processing

Message processing happens in cascading steps,

When a message (or a series of messages) is *pushed* to Event Connector, it is validated to ensure that the data is correctly formatted and passes the required access authorization.

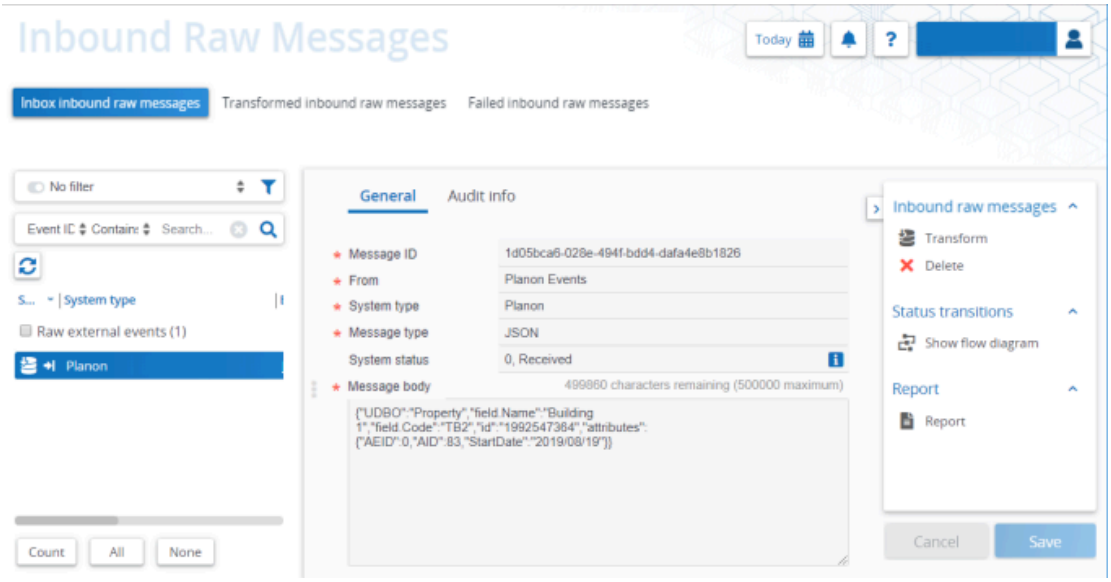
The body of raw messages cannot exceed 32,000 characters.
If this limit is exceeded, an error will be displayed when trying to create the raw message (*inbound* or *outbound*).
If customers want to truncate messages that are too long, this should be handled in a customized app.

Inbound transformation

Messages are received and put on the queue.

Inbound

Raw messages are displayed on the Inbound raw messages > Inbox inbound raw messages step.



Here, they await transformation. During transformation, the Planon application will find the corresponding transformers based on the message type and system type combination.

If there are multiple transformers for the same combination, they will be picked randomly.

Once transformed, the status of the message is changed to *Transformed* and is shown in the **Transformed inbound raw messages** step.

If transformation fails, the message will be listed on the **Failed inbound raw messages** step. The application will put the message back on the queue for processing with a maximum number of 9 attempts before it is finally deemed as *Failed*.

Outbound transformation

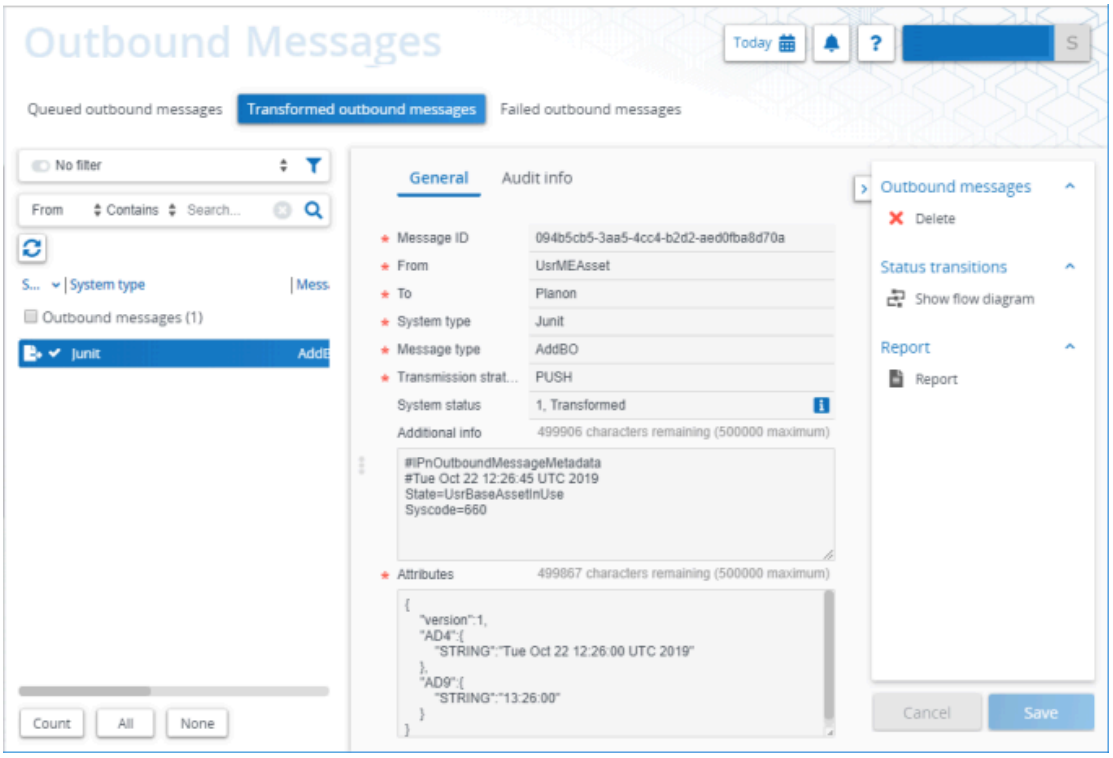
Messages are received and put on the queue.

Outbound

Messages are displayed on the Outbound messages > Queued outbound messages step.

Here, they await transformation. During transformation, the Planon application will find the corresponding transformers based on the message type and system type combination.

If there are multiple transformers for the same combination, they will be picked randomly.

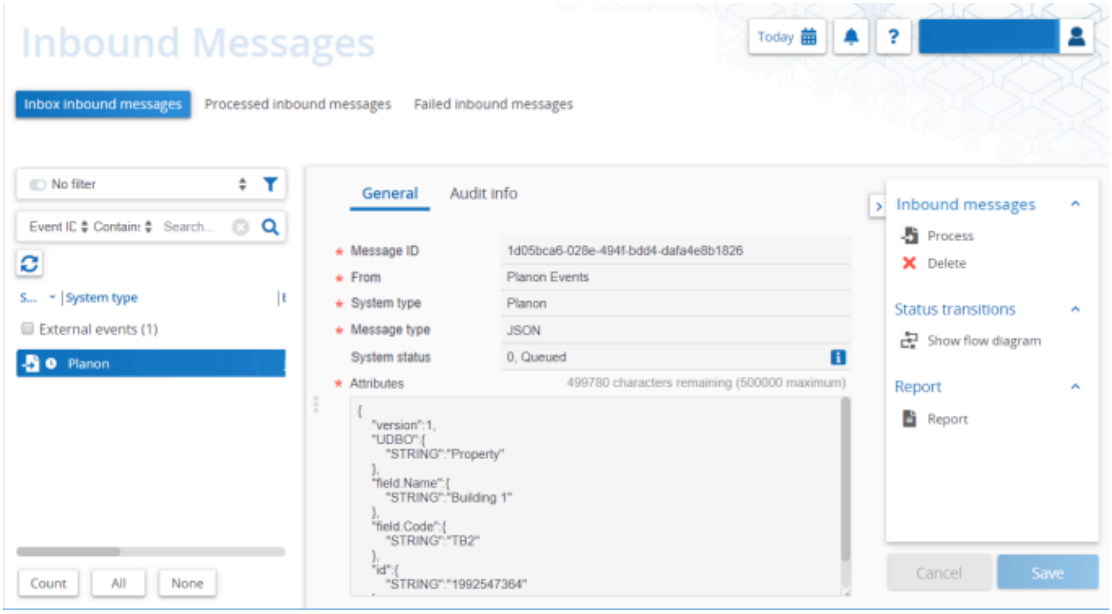


Once transformed, the status of the message is changed to *Transformed* and is shown in the **Transformed outbound messages** step.

If transformation fails, the message will be listed on the **Failed outbound messages** step. The application will put the message back on the queue for processing with a maximum number of 9 attempts before it is finally deemed as *Failed*.

Processing

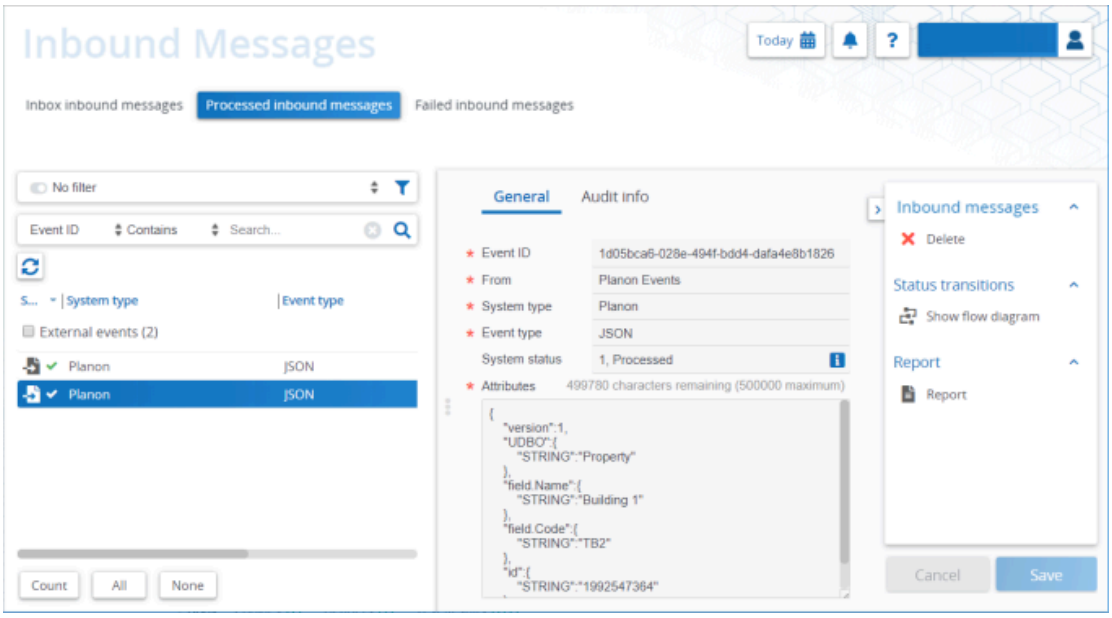
After transformation, the messages are displayed in the Inbound messages > Inbox inbound messages step.



Here, once more, it is put on the queue and await processing. During processing, the Planon application will find the corresponding processors based on the message type and system type combination.

If there are multiple processors, all will be executed.

Once done, the message status is changed to *Processed* and it proceeds to the **Processed inbound messages** step.



If processing fails, the message will be listed on the **Failed inbound messages** step. The application will put the message back on the queue for processing with a maximum number of 9 attempts before it is finally deemed as *Failed*.

Reprocessing stuck messages

When Planon is restarted all queued messages are lost and cannot be processed.

If you want messages to be processed, they must be put on the queue again.

To queue and process messages again, click the **Recreate** action, which will ensure that these so-called stuck messages are put back on the queue again.

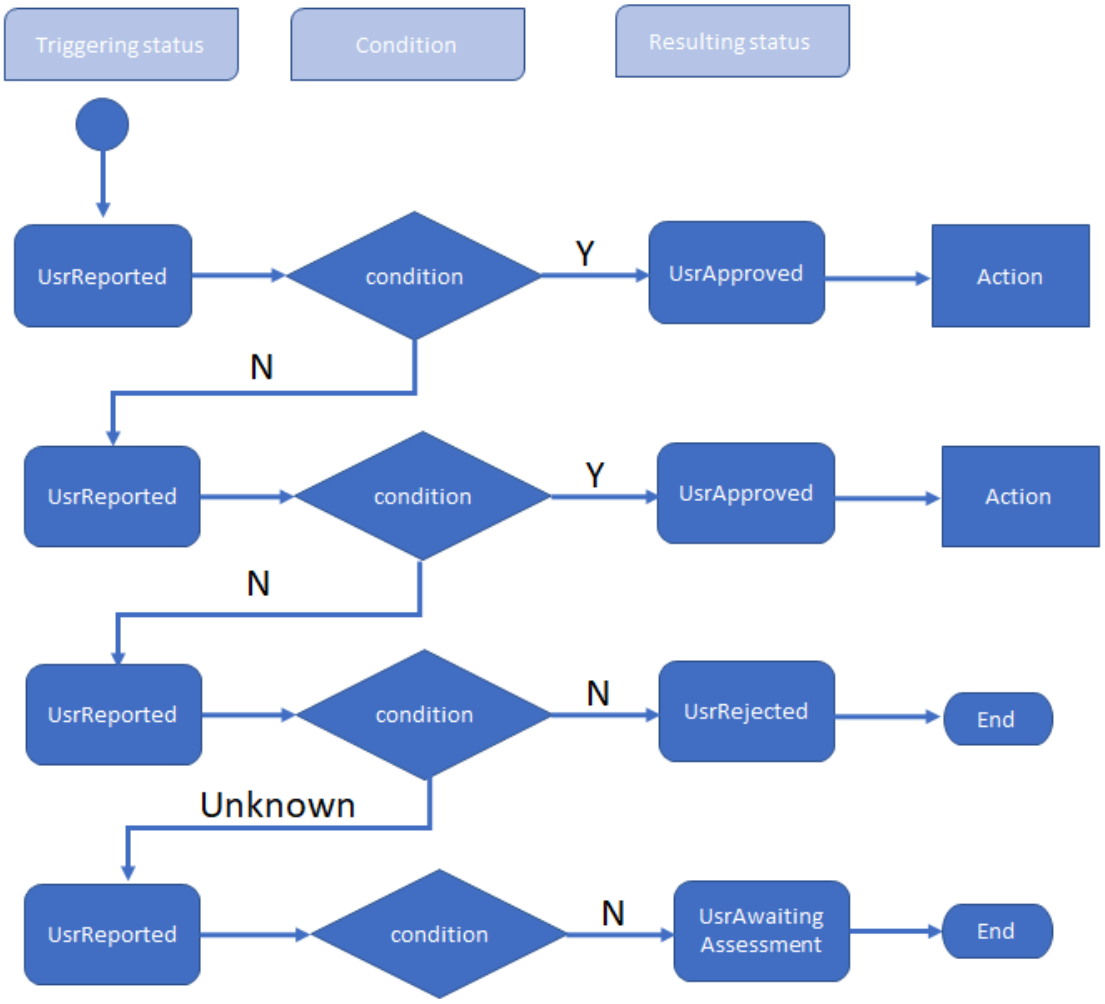
The action is available on the **Queued messages** step. Triggering this action first shows a warning message. When proceeding, this will remove all selected messages and put them back on the queue again.

It is possible to recreate **ALL** messages, but this can lead to an error if the message that is selected for recreation is already being processed. We recommend, therefore, to only do this for messages that are on the queue for a longer period of time.

Business events

After processing, the Edition appropriate actions are carried out.

Events are processed in the Planon application by submitting them to a set of business rules. The decision model kicks into action when adding a business event or setting a triggering status.

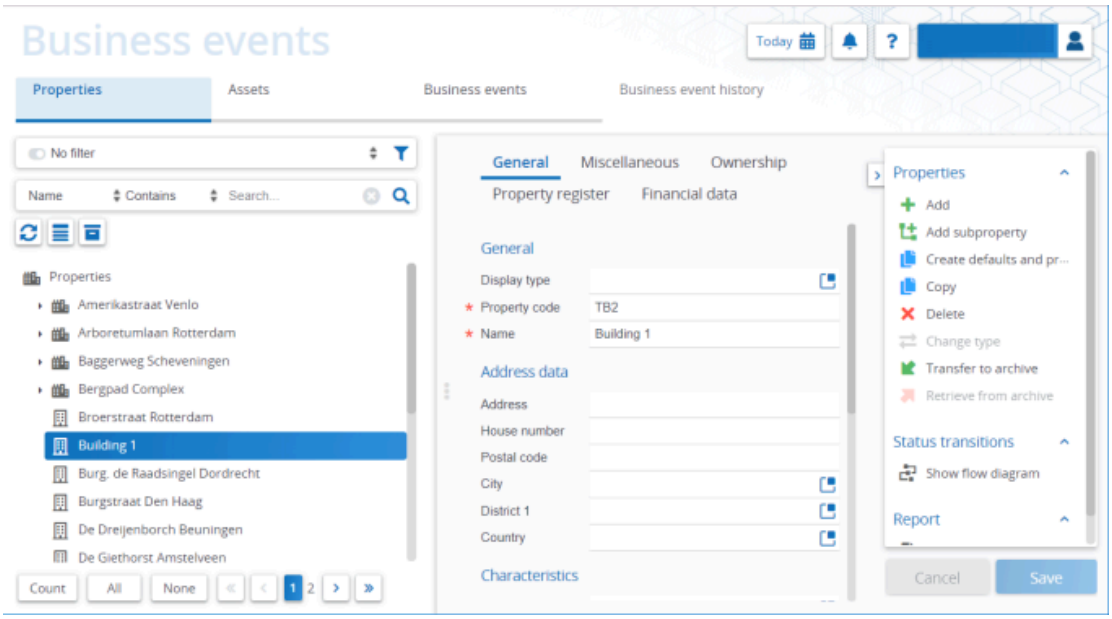


The whole model of rules is subsequently followed through until a condition is met. This subsequently results in the action that is defined for the rule. If no condition is met, the model simply stops for manual assessment.

A condition is not mandatory, in which case it is always true.

Examples of actions could be:.

- Creating or updating a property (**Properties** step).
- Creating or updating an asset (**Assets** step).
- Creating or updating a business event (**Business events** step), sending a message to an external system.



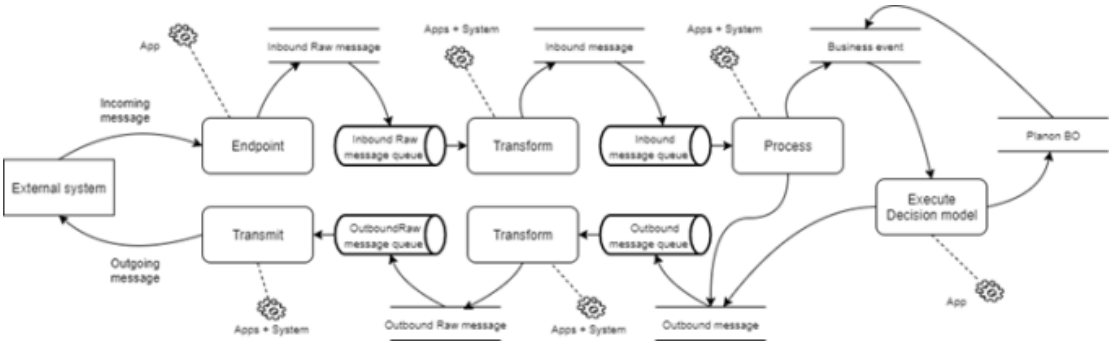
Event connector

Event connector is an infrastructure in Planon that is used to create a reliable exchange of messages between a Planon application and another application.

The event connector is NOT an *out-of-the-box* product, it creates the infrastructure to receive or send messages and to process them. The actual execution of a step in the process must often be provided by an app (Tailor Made Software).

An app can be created using the Planon app development IDE. The IDE, and examples provided by Planon, make it easy to develop, test and deploy an app. An app can also be developed by Planon Platform Solutions Expertise Center, the customer or any third party.

The next image shows an overview of the event connector infrastructure in Planon. It depicts how inbound message and outbound messages are processed.



With the event connector infrastructure, any kind of message can be handled and the effect of the execution is implemented by the app. An app must provide the following parts:

- Inbound event
- Outbound event

Event connector

Inbound event

Endpoint: Event connector can handle any kind of message.

This is only possible because the endpoint must be provided by an app. The endpoint then transfers the inbound message to an *Inbound Raw Message*.

The raw message contains the original message as it was received.

Transform: The Transform step transforms the *Inbound Raw Message* to an *Inbound Message*. If the message that was originally received was in a structured form, than this step can just pass the message. If the incoming message is in a free format style, than it must be structured.

Process: The Process step is the point where the message is actually handled. In the Process step, the message can be directly handled, or it can be passed to the decision model. If it is directly handled, the app should directly execute actions on the Planon business objects. The decision model is a way to handle the event by configuration, it is optional.

Outbound event

Business rule: The business rule is the point where a message can be created and be passed to Event connector.

Business rules can be developed with the Planon IDE and it must be linked to an event of the Business Object (BO). When the BO is saved, the business rule is executed. The Planon Platform provides an API to access Event connector from a Business rule and thus to make it easy to create and send messages from a Business rule.

Transform: The Transform step transforms the *Outbound Message* to an *Outbound Raw Message*. If the message that was originally received was in a structured form, then this step can just pass the message otherwise it has to be reshaped to a form that is suitable for the receiver.

Transmit: The Transmit step actually sends the message.

- The infrastructure provides just the basic handling of the message. It controls the flow of messages, makes it reliable and controllable. It is equipped to handle errors during processing etc.
- The framework does not perform any actions in the Planon business model itself. The implementation of the app is essential. In each step,

the implementation can check the content of the message and decide whether it should be processed.

Endpoint security

The endpoint receives the inbound messages.

The event connector provides a generic endpoint. This generic endpoint is only used for system messages, messages created by a Planon software.

Messages created by other applications must be sent to an endpoint provided by an app.

An endpoint will only receive a message when the right credentials, user name / password or access token are provided. From that perspective, access to the event connector endpoint works like any other REST (JAX-RS) endpoint.

Message processing security

An app provides the components / plug-ins that actually handle the messages.

Therefore, a plug-in must subscribe to the message types. Internally, the framework uses the namespace to ensure that apps subscribe only to their own messages.

Field descriptions

The following section(s) describe(s) the fields, their purpose and meaning.

(Raw) message fields

The (raw) message fields contain the metadata used for identifying its source and processing attributes. These fields are the same for both the inbound and outbound channels.

Property	Description
messageID	The identifier that uniquely identifies the message. If the same message is sent twice, which can happen if a network problem occurred, Planon will use this identifier to detect the duplicates. The identifier must be a UUID.
From	A unique identification of the message's sender, unique in the context of the Planon application.*
systemType	A unique identification of the type of the system. The systemType value is used to match the transformer and processor to the message.* For connectors in an app, the systemType should always start with the group name of the app.
messageType	An identification of the type of the message. The messageType must be unique in the context of the systemType. The messageType value is also used to match the transformer and processor to the message.*
messageBody	Contains the actual message content. The only requirement is that it is a valid JSON value and does not exceed 32,000 characters. This value will be passed to the processor.
transmissionStrategy	Lists the strategy that is used (push/pull) to deliver the message to the external system.
Attributes	Contains the transformed message in the form of key value pairs.
Remark	Contains remarks registered when processing the message. Contains details when a failure occurs.

Property	Description
Additional info	Additional info that cannot be in the message body and that is needed to send the message. This could, for example, be mail settings or endpoint information. Event connector is designed as a light-weight messaging system. Therefore, the maximum size of the **Additional info** is 64Kb.

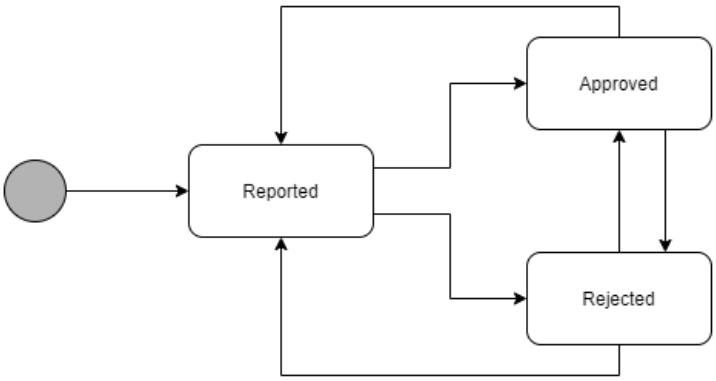
*Do not add leading or trailing spaces, these will be removed.

Raw message

Business event fields

In addition to the regular Planon fields for **Properties** and **Assets**, **Business events** consists of the following fields.

Field	Description
Asset	The asset for which this business event is created, if applicable.
Attribute set definition	Select the attribute definition set for this business event. By selecting an attribute definition set you can expand the number of properties to be captured for an entity. When applying an attribute set, the specified fields will appear in the data panel. For more information see System Management > Attributes.
Inbound message	A reference to inbound message for which this business event is created.
Name	Here, you can specify a name for the business event.
Property	The property for which this business event is created, if applicable.
User-defined business object	Displays the user-defined business object for which the business event was created.
System status	Indicates the status of the business event:

Field	Description
	• Reported The initial status in which the business event is created. The business event has not yet been processed. • Approved The business event has been processed and approved by the system. • Rejected The business event has been processed and rejected by the system. 
User-defined status	Displays the corresponding user-defined statuses specified for the **Business events** BO. The decision model will only be executed when changing a user-defined status, a so-called *triggering status*.
Remark (1-3)	Three fields for storing a large body of text.
URL (1-2)	Two fields for storing URLs. These type of fields cannot be stored in the **Remark** fields.
Decision model	Select the decision model to be applied. When a decision model is available, it will be triggered when a new business event is created.
Failed?	Indicates if the execution of the linked decision model has failed.
Error log	If an error occurred during the execution of a decision rule, the error will be shown here.

Business event

Decision models/rules - fields

Field	Description
Code	A code to identify the decision model.
Name	A name to identify the decision model.
Remark	Free-definable field, you could e.g. include a description of the decision model.
Decision rules	
Sequence	Enter a value for processing the rule. A decision model can contain various rules, they will be executed in the order defined here (1 being the first rule to be executed).
Decision model	Select the decision model to which you want to link the rule.
Triggering status	The user-defined intial status that triggers the condition rule.
Condition	Specify the filter criteria to be verified. By using And/Or operators you can expand the filter criteria. • If the condition is true, the next rule will be executed. If there are no more rules to be executed, the required action is carried out. • If the condition is false, the process stops and the decision model status changes to *UsrRejected*. • If the condition cannot be determined, the status of the decision model changes to *UsrAwaitingAssessment*.
Action worker	Select the action (business rules) to be executed once the conditions are met. The business rules are available in the **Apps** TSI (and constructed in the *Planon Platform IDE*). If the field is empty then only the status of the business event will be set (via the **Resulting status** field)
Resulting status	The status in which the rule will be set once the condition has been verified (**UsrApproved** or **UsrRejected** **UsrAwaitingAssessment**)

Field	Description
Partner identifier	The namespace (PartnerID, App name, Module name) is used to select and find the class in a decision rule.
App name	
App module name	
Remarks	Free-definable field, you could e.g. include a description of the decision rule.
Settings tab	
Settings	The settings that apply to the decision rul (JSON format). The user has to define the values of the various settings in this field. When the app is upgraded (introducing new or changing settings), this field is not automatically changed. When this happens, you must manually update this field.
Settings example tab	
Settings example	A calculated field with example of the settings, in JSON format. This is intended to help define the correct settings in the **Settings** field. When the app is upgraded (introducing new or changing settings), this field is updated according to the new settings.
Settings schema tab	
Settings schema	A calculated field with the type/format and constraints that apply to the settings, in JSON format. When the app is upgraded (introducing new or changing settings), this field is updated according to the new settings.

Decision model

Index