Application Help for Production Process Designer

Generated on: 2025-12-14 10:49:36 GMT+0000

SAP Digital Manufacturing | latest

Public

Original content: https://help.sap.com/docs/SAP_DIGITAL_MANUFACTURING/f1949e1d1fd24d7e998e5f70fecc1835?locale=en-US&state=PRODUCTION&version=latest

Warning

This document has been generated from SAP Help Portal and is an incomplete version of the official SAP product documentation. The information included in custom documentation may not reflect the arrangement of topics in SAP Help Portal, and may be missing important aspects and/or correlations to other topics. For this reason, it is not for production use.

For more information, please visit https://help.sap.com/docs/disclaimer.

About Production Process Designer

The Production Process Designer helps you model production processes and get transparency when you create the layout of your shop floor. Production processes define the interaction between machines or rules, actions, and workflows that control the execution on the shop floor.

Advantages of Production Process Designer

With the Production Process Designer, you can:

- Model production processes using a visual designer
- Focus on business logic instead of on programming skills
- Manage configurations centrally instead of programming each piece of equipment individually
- Manage different versions of configurations
- Adapt configurations easily if the environment changes

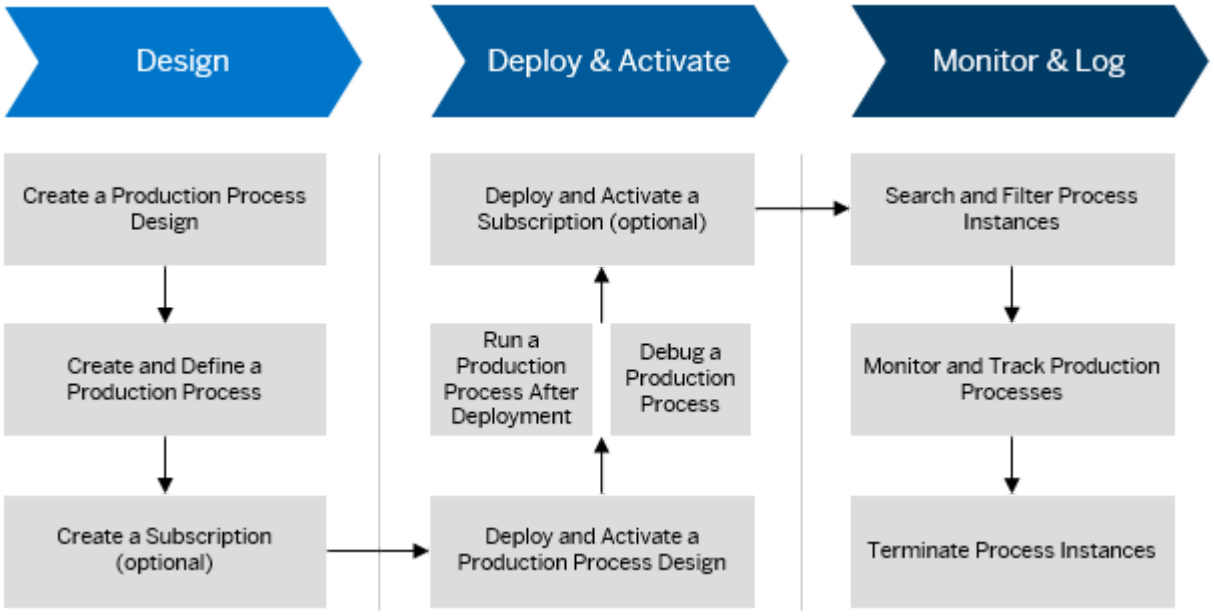
Related Information

Production Process Administration
Design Production Processes

Workflow Overview

The following diagram shows the general workflow to use production process:

This image is interactive. Hover over each area for a description. Click highlighted areas for more information.



Please note that image maps are not interactive in PDF outputs.

Scenarios

With the production process designer, you can design production processes that fit for different scenarios. For each scenario, you need to prepare some master data beforehand. The following sections explain what master data must be created for each scenario.

Scenario 1: Machine Automation

To control machines without human intervention, design an automation sequence. The machine-to-machine or machine-to-system communication is orchestrated by a Production Connector system.

For this scenario, you need to prepare the following master data:

- A Production Connector system
- One or more external service providers with **Production Connector / Plant Connectivity** as the destination type, associated with the above Production Connector system

  These service providers provide shop floor services (OPC UA method) which can be triggered by the automation sequence, for example, to start a machine operation.
- One or more third-party services

  You need to connect the Production Connector web server automatically created in the **Manage Web Servers** app with other related web servers and add the web services you would like to use. For details, see Manage Web Servers.

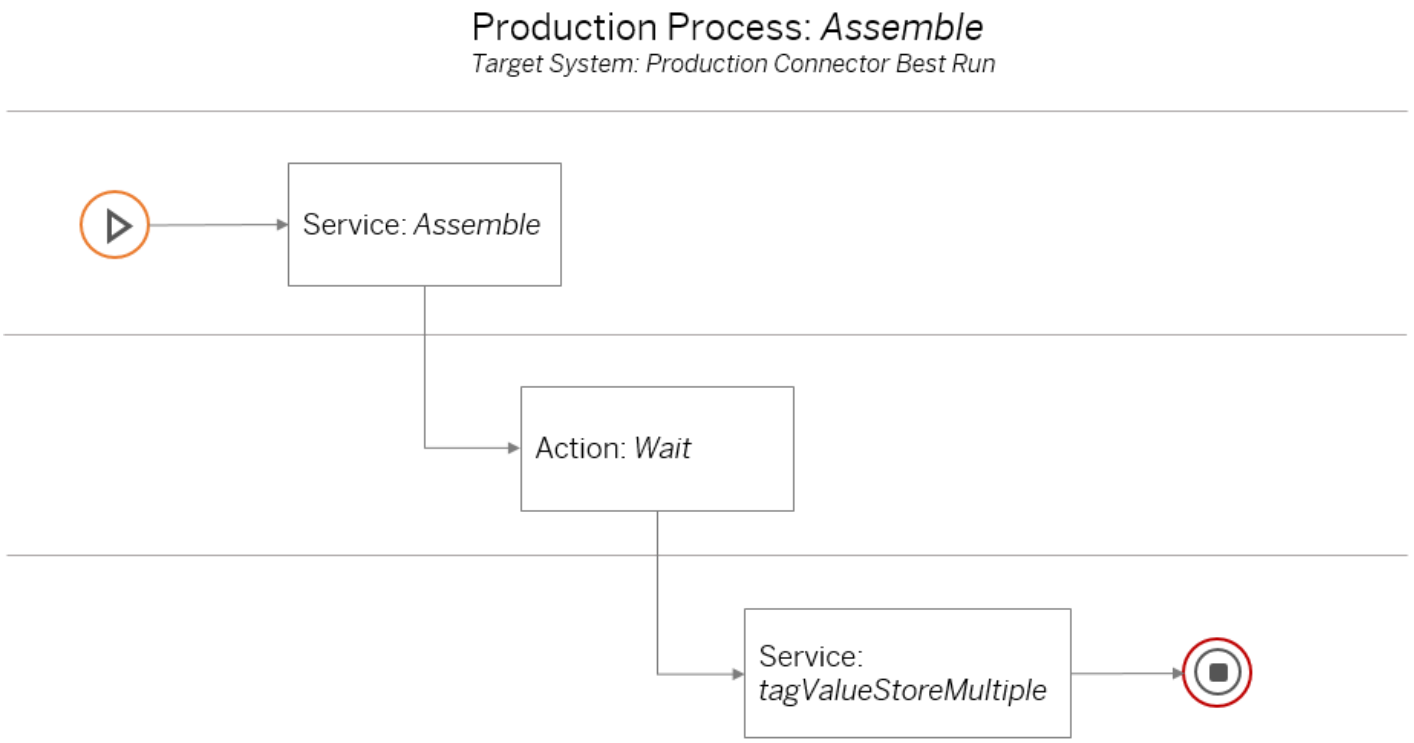
Example

Create a production process to use a robot to assemble a product. After a short pause, write values to some indicators. You can also define a conditional branch to cover error handling in the process.

When the production process is started, the shop floor service (OPC UA method) `Assemble` provided by the external service provider `Robot A` is called. This service performs some assembly activities, for example, assembles a bike.

After one second, the production connectivity model service `tagValueStoreMultiple` provided by the Production Connector web server is called and the output values of the `Assemble` service are stored in some tags.

The interval of one second is defined by the `@Wait` function.



Scenario 2: Manufacturing Execution Automation

To manage and control manufacturing and shop floor operations, design a cloud process. By enabling this kind of production process, you can automate activities (such as start an SFC) or enforce activities (such as data collection) on the shop floor.

For this scenario, you need to prepare the following master data:

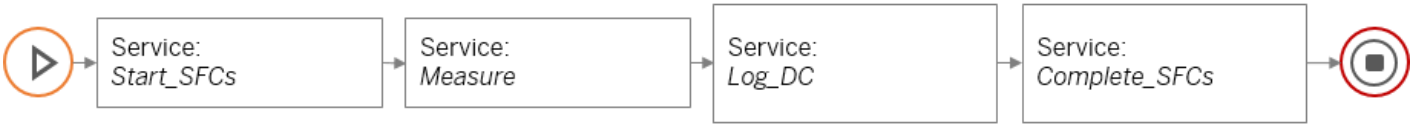
- One or more web services

  You need to connect the SAP Digital Manufacturing service type web server automatically created in the **Manage Web Servers** app with other related web servers and add the web services you would like to use. For details, see Manage Web Servers.

Example

- Start an SFC at an operation activity on a particular resource (service: `Start_SFCs`)
- Measure object (service: `Measure`)
- Collect data from the shop floor (service: `Log_DC`)
- Complete the SFC (service: `Complete_SFCs`)

For more information about these services, refer to Available Services and Subprocesses.



Scenario 3: Convergence of Manufacturing Execution and Machine Automation

In the third scenario, you combine scenarios 1 and 2 to embed an automation sequence inside a cloud process.

Create a production process that is run in the cloud and another production process (automation sequence) that is run by a Production Connector system. Then embed the second in the first.

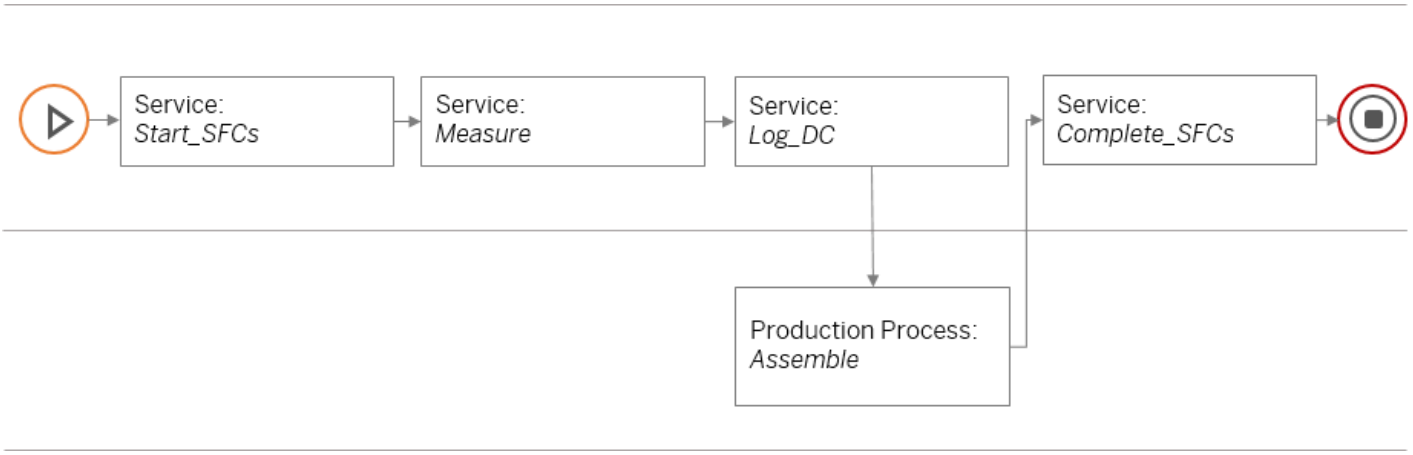
Note

To embed a production process in another production process, both production processes must be included in the same production process design.

For this scenario, you need to prepare the master data required for both scenario 1 and scenario 2.

Example

Scenario 3: Compared with the example for scenario 2, after starting an SFC, a robot assembles the product.



Similarly, you can also embed a cloud process inside an automation sequence.

Related Information

Tutorial: Automation with Production Process Design

Prerequisites

There are some prerequisites to use the Production Process Designer.

Create a Destination for SAP Digital Manufacturing

In an automation sequence or a subscription, to call SAP Digital Manufacturing cloud services or cloud processes defined in the Production Process Designer, create a destination for SAP Digital Manufacturing in the SAP BTP cockpit.

Prerequisites

- You are a member of the SAP Business Technology Platform global account.
- You have obtained the service key information of an SAP Digital Manufacturing service instance. For more information, see Prepare for API Integration.

Context

SAP Digital Manufacturing cloud services are API services provided by SAP Digital Manufacturing. For details, see SAP Business Accelerator Hub.

The actual services available may be more than those listed under the public API packages in SAP Business Accelerator Hub. There are some internal services as well. You can check all services under DMC group in the **Manage Service Registry** app.

Procedure

1. Go to your subaccount.
    a. Log on to the SAP BTP cockpit and select a region.
    b. Select a global account.
    c. Select a subaccount that you're working on for production process.
2. Choose **Connectivity** > **Destinations**.
3. Choose **Create Destination**.
4. Specify the destination information as follows:

Property	Value
Name	`SAP_DMC_DEFAULT_SERVICE_KEY`
Type	`HTTP`
Description	Not required
URL	value of `public-api-endpoint` in the service key Note that this URL does not refer to the authentication endpoint. A typical URL, for example, should look like `https://api.xxxx.dmc.cloud.sap`.
Proxy Type	`Internet`
Authentication	`OAuth2ClientCredentials`
Client ID	value of `uaa.clientid` in the service key
Client Secret	value of `uaa.clientsecret` in the service key
Token Service URL	Enter the `uaa.url` in the service key and add `/oauth/token` to the end. A typical token service URL, for example, should look like `https://{$Subdomain}.authentication.xxxx.hana.ondemand.com/oauth/token`
Token Service User	Not required.
Token Service Password	Not required.

Note

You can find the `uaa.clientid`, `uaa.clientsecret`, and `uaa.url` in the service key of service instance by choosing **Instances and Subscriptions** → Select an instance → **Service Keys**. See "Create service key" section in Prepare for API Integration for more information.

The services can have updates upon new releases. If you come across errors when calling the services, you have to re-create the service key and destination.

5. Save the destination.

Configure Shop Floor Services (OPC UA Methods)

Shop floor services (OPC UA methods) are services provided by external service providers that are created on the selected runtime Production Connector system.

Context

You need to configure these services in the **Manage Service Providers** app first to use them in the automation sequence and subscription. To do so, follow the steps below:

1. Open the **Manage Service Providers** app and choose *(Create New Service Provider)*.
2. Input the name and description of the service provider.
3. Choose **Production Connector / Plant Connectivity** as the destination type.
4. Choose **External** as the service provider type.
5. Select the Production Connector system.

   Production Connector systems configured in the Configure Production Connectivity app are available for selection.
6. Choose **OPCUAServer** as the Production Connector usage type.
7. Configure the maximum number of retry attempts and retry interval.
8. Configure the **Data Source Properties** and **Security** section.

   See Step 6 in Configuring Service Provider for Services-Based Communication for details.
9. Choose **Save**.
10. Configure services in the service provider.

    See Configuring Services in Service Provider for details.

> **Note**
>
> You can find a documentation on the supported OPC UA data types in SAP Note 3461876.

A client proxy contains the authentication details required when the service receives a call from the Production Connector. A client proxy is created as a destination in the Production Connector.

To create a client proxy:

1. In the **Manage Service Providers** app, select the service provider you have created.
2. Choose the **Service** tab.
3. Select the service group and choose a service.
4. Choose **Client Proxy Configuration**.
5. Choose *(Add)* to add a client proxy.

6. Input the name and description of the client proxy.

7. Choose **Submit for Deployment** if you want to add the client proxy to a deployment group.

   For details, see Configuring Services in Service Provider.

> **Note**
>
> You can edit or delete a client proxy only if it is in the **Draft** status. If you want to edit a client proxy that is in the **Deployed** status, you must first remove the client proxy from the deployment group. For more information on deployment groups, see Deploy Shop Floor Elements.

Configure Client Proxy for Third-Party Service

To use a third-party service, SAP Digital Manufacturing cloud service, SAP Digital Manufacturing for edge computing service, or registered cloud/edge process in an automation sequence or subscription, the service must have a client proxy configured for it. When defining the automation sequence or subscription, you either need to create a new client proxy or select an existing client proxy for this service.

Prerequisites

For third-party service:

- You have defined the service metadata in the **Manage Service Registry** app. For details, see Create an API Service (RESTful and OData).
- You have created a web server in the **Manage Web Servers** app. For details, see Manage Web Servers.
- You have assigned the services with web server in the **Manage Service Registry** app. For details, see Create an API Service (RESTful and OData).

The third-party service, called by an automation sequence or subscription of Production Connector on Windows runtime, supports the following authentication methods:

1. No authentication (anonymous)
2. Basic authentication
3. OAuth 2.0 Client Credentials authentication
4. X.509 certificate authentication

- For authentication methods 2 and 3: You have created a destination in SAP BTP cockpit first and then added the destination, connected the Production Connector runtime web server with the web server you have created, and added and deployed services in the **Manage Web Servers** app. For details, see Create HTTP Destinations and Manage Web Servers.
- For authentication method 4: You have connected the Production Connector runtime web server with the web server you have created, selected the service certificate, and added and deployed services in the **Manage Web Servers** app. For details, see Manage Web Servers and Adding and Deploying Services from a Connected Web Server.

For SAP Digital Manufacturing cloud service or registered cloud process, called by an automation sequence or subscription of Production Connector on Windows runtime:

- You have created a destination in SAP BTP cockpit. For details, see Create a Destination for SAP Digital Manufacturing.

- You have connected the Production Connector runtime web server with the SAP Digital Manufacturing services type web server and added the services or processes you want to use from the connected web server. For details, see Manage Web Servers and Adding and Deploying Services from a Connected Web Server.

For SAP Digital Manufacturing for edge computing service or registered edge process, called by an automation sequence or subscription of Production Connector on edge runtime:

- You have added the services or processes you want to use from the connected web server. For details, see Adding and Deploying Services from a Connected Web Server.

Context

For any client proxy for a third-party service, some configurations are predefined in the **Manage Service Registry** app for the service. For example, the endpoint URL and headers.

Procedure

1. Edit a production process.
    a. Open the **Design Production Processes** app.
    b. Open a production process design.
    c. Select a production process with a Production Connector runtime environment.
2. In the production process editor, drag and drop a third-party service, SAP Digital Manufacturing cloud service, or registered cloud process to the canvas.
3. On the right panel, choose the link next to **Client Proxy**.
4. In the **Select a Client Proxy** window, select an existing client proxy or create a new client proxy.

    Choose **From Destination** to create a client proxy from the destination that you added to the web server.

    Choose **From Certificate** to create a client proxy from the certificate that you added to the web server.

    Choose **Anonymously** to create a client proxy if the service has no authentication (anonymous).

    > **Note**
    > To use an SAP Digital Manufacturing cloud service or registered cloud process in an automation sequence or subscription of Production Connector on Windows runtime, choose the link next to **Client Proxy** and select `SAP_DMC_DEFAULT_SERVICE_KEY` as the client proxy destination.
    >
    > To use an SAP Digital Manufacturing for edge computing service or registered edge process in an automation sequence or subscription of Production Connector on Windows runtime, choose the link next to **Client Proxy** and create a client proxy from the certificate. For details on the certificate, see Enable Production Connector to Invoke Edge Services.
    >
    > To use an SAP Digital Manufacturing for edge computing service or registered edge process in an automation sequence or subscription of Production Connector on edge runtime, choose the link next to **Client Proxy** and choose **Create** to create a new client proxy.

5. Save your entries.

Use Third-Party Service in Cloud/Edge Process

You can use third-party service in a cloud/edge process, timer, or business rule.

Prerequisites

- You have defined the service metadata in the **Manage Service Registry** app. For details, see Create an API Service (RESTful and OData).
- You have created a web server in the **Manage Web Servers** app. For details, see Manage Web Servers.
- You have assigned the services with web server in the **Manage Service Registry** app. For details, see Create an API Service (RESTful and OData).

ℹ Note

Self-assigned server certificate (HTTPS) is not supported for third-party services called in cloud or edge processes.

If you use XSUAA (Extended Services for User Authentication and Authorization), set the access token validity of your token servers longer than the execution times of your production processes.

Third-Party Service in Cloud Process

The third-party service, called by a cloud process, timer, or business rule, supports the following authentication methods:

1. No authentication (anonymous)
2. Basic authentication
3. OAuth 2.0 Client Credentials authentication (cloud services only)
4. OAuth 2.0 SAML Bearer Assertion authentication (not supported when the third-party service is called by a timer or business rule)

   ### ⚠ Caution

   When you use this method for a third-party service in the cloud process, make sure you don't use an automatic trigger or automation sequence to call the process. Otherwise, you'll encounter errors.

   When you use methods 1, 2, or 3 for a third-party service in the cloud process, make sure you don't change the authentication method in the destination to OAuth 2.0 SAML Bearer Assertion after deploying the process. Otherwise, you need to redeploy the process, including its parent process.

- For authentication methods 2, 3, and 4: You have created a destination in SAP BTP cockpit first. Then, you have added the destination and connected the cloud runtime web server with the third-party service web server you have created in the **Manage Web Servers** app. For details, see Create HTTP Destinations and Manage Web Servers.

Third-Party Service in Edge Process

The third-party service, called by an edge process, supports the following authentication methods:

1. No authentication (anonymous)
2. Basic authentication

   ### ℹ Note

   The following keys are required for each secret's data:

   - URL
   - Authentication (value: `BasicAuthentication`)
   - User
   - Password

All the values should be base64-encoded.

3. OAuth 2.0 Client Credentials authentication

> **Note**
>
> The following keys are required for each secret's data:
>
> - URL
> - Authentication (value: `OAuth2ClientCredentials`)
> - clientId
> - clientSecret
> - tokenServiceURL
>
> All the values should be base64-encoded.

- For authentication method 2 and 3: You have to maintain the secret information in a Kubernetes secret in the SAP Digital Manufacturing for edge computing namespace first, and connect the edge runtime web server with the third-party service web server you have created in the **Manage Web Servers** app. For details on connecting the web server, see Manage Web Servers.

  There are two ways to maintain the secret information in the Kubernetes secret:

  - Manually provision the Kubernetes secrets.

    1. For basic authentication, create a YAML file, for example, test-basic.yaml like the following:

       ```
       apiVersion: v1
       data:
         Authentication: QmFzaWNBdXRoZW50aWNhdGlvbg==
         Password: base64-encoded-password
         URL: base64-encoded-url
         User: base64-encoded-user
       kind: Secret
       metadata:
         name: test-basic
       type: Opaque
       ```

       For OAuth 2.0 Client Credentials authentication, create a YAML file, for example, test-oauth2.yaml like the following:

       ```
       apiVersion: v1
       data:
         Authentication: T0F1dGgyQ2xpZW50Q3JlZGVudGlhbHM=
         URL: base64-encoded-url
         clientId: base64-encoded-client-id
         clientSecret: base64-encoded-client-secret
         tokenServiceURL: base64-encoded-token-service-url
       kind: Secret
       metadata:
         name: test-oauth2
       type: Opaque
       ```

    2. Create a Kubernetes secret to store the secret information:

[basic authentication]:

```
kubectl apply -f test-basic.yaml -n [dm_edge_namespace]
```

[OAuth 2.0 Client Credentials authentication]:

```
kubectl apply -f test-oauth2.yaml -n [dm_edge_namespace]
```

- Automate the Kubernetes secrets lifecycle by leveraging the External Secrets Operator. For details, see Tutorial: Leverage External Secrets to Automate Kubernetes Secret Creation and Update.

Note

Only a production process of SAP Digital Manufacturing for edge computing (version 2505 or higher) can call the third-party service.

Procedure

1. Edit a production process.
   a. Open the **Design Production Processes** app.
   b. Open a production process design.
   c. Select a production process of cloud/edge runtime environment.
2. In the production process editor, drag and drop a third-party service to the canvas.
3. If you use a third-party service in the cloud process, on the right panel, choose the destination you have previously added to the web server.

   If you use a third-party service in the edge process, on the right panel, input the name of Kubernetes secret. If no authentication is used, leave the field blank.

Restrictions for Cloud Processes

See the related section in 3379404 for details.

Design Production Processes

Use the **Design Production Processes** app to create production processes in different production process designs.

With the Production Process Designer, you can model production processes and gain transparency when creating the layout of your shop floor. A production process can define the interaction between machines, define rules, actions, and workflows that control the execution on the shop floor, or a mixture of both.

When you deploy and activate production process designs, the system translates processes into configuration and stores them in the corresponding runtime environments.

A production process design can have more than one version. Any changes to the configurations require a new version. A user must be assigned to a production process design to edit, deploy, or delete it. The production process design can be fully edited in Draft or Modified status, but is partially editable in Failed status. It is validated when being deployed. Once deployed, it can no longer be changed or deployed again. To update a production process design that has been deployed, create a new version.

Production process designs can have the following statuses:

- **Draft**: The production process design has just been created or edited before being submitted to the deployment group.
- **Modified**: A new version of a deployed production process design has been saved.
- **Awaiting Deployment**: The production process design has been submitted to the deployment group but not deployed.
- **Deployed**: The production process design has been sent to the runtime environment.
- **Awaiting Un-deployment**: The deployed production process design has been submitted to a new deployment group for deletion.
- **Archived**: The production process design has been deleted from the runtime environment.
- **Failed**: Deployment failed and you need to try again to deploy it.

Production processes can have the following statuses:

- **Editable**: The production process has just been created or edited before being submitted to the deployment group or it has been submitted to the deployment group but failed to be sent to the runtime environment.
- **Deployed**: The production process has been sent to the runtime environment.
- **Archived**: The production process has been deleted from the runtime environment.

ℹ Note

You can save your search with filters by applying a smart variant. Smart variant is supported by the **Design Production Processes**, **Deploy Shop Floor Elements**, **Manage Automatic Triggers**, **Monitor Production Processes**, **Monitor Event Business Rules**, **Production Process Administration**, **Recover Production Processes**, and **Manage Service Registry** apps. To apply a smart variant, choose the □ *(Select View)* next to the **Standard** text in the header of the list page and then choose **Save As**. Add a name for the variant.

For an example of creating a production process in the **Design Production Processes** app, see Tutorial: Automation with Production Process Design and Creating a Printing Production Process (Version 1).

ℹ Note

The Production Process Designer uses the time zone of your selected plant as the standard time. If no plant is selected, the time of your current browser will be displayed instead.

ℹ Note

Although all apps are accessible on desktop, tablet, and mobile, we strongly recommend using them on the desktop where you have the best user experience. Please note that some features of the **Design Production Processes** app may be inapplicable on tablet and mobile.

Concepts

A few concepts and background information for the Production Process Designer before use.

Available Services and Subprocesses

In the **Design Production Processes** app in SAP Digital Manufacturing, different types of production processes use different services. Additionally, a process can embed other processes as subprocesses for more flexible reuse.

ℹ Note

In the service library, services with web server assigned are grouped as root node while services without web server assigned are put under one independent folder. If the service is not assigned with any web server, it cannot be used in the process.

To use snippet, see Create and Use Snippets.

A parent process can never nest in a subprocess. For example, `Process A` → `Process B` → `Process A` is not allowed.

Cloud runtime environment

Process Runtime Web Server	Web Server to Connect	Object Included	Note
SAP Digital Manufacturing services type web server	(not required)	SAP Digital Manufacturing cloud services	[1]
		[Production Connector on Windows] read/write indicators (built-in services)	[15], [2]
		read/write indicators locally (built-in services)	[3]
		read/write indicators flexibly (built-in services)	[18]
		read/write global variables (built-in services)	[20]
		array/map services (built-in services)	[28]
		POD plugins	[4]
		registered cloud processes	[5]
		cloud processes (subprocess)	[19]
		[Production Connector on Windows] automation sequences (subprocess)	[15], [8]
	user-defined web server (cloud)	third-party services (RESTful, OData or SOAP)	[6], [16], [7]
	user-defined web server (on-premise)		

Production Connector runtime environment

You should have configured Production Connectivity Model with the shop floor system added first before creating automation sequences. For details, see About Production Connectivity Model.

In addition, to set up connection with the Production Connector on edge runtime web server, you should have installed SAP Digital Manufacturing for edge computing, deployed the **Production Process Orchestration**, **Production Connectivity Model**, **Production Connector** module, performed certain configurations in Cloud Connector, and configured Production Connector on edge listed in the **Configure Production Connectivity** app. For details, see Install, Upgrade, and Uninstall, Modular Deployment, Enable Deploying Automation Sequences of Production Connector on edge Runtime and Updating a Production Connector on Edge Object.

You can also expose the automation sequence as a RESTful service to be called by machines. For details, see Expose Automation Sequence as RESTful Service.

ℹ Note

In some cases, built-in services in the left panel may be unavailable for automation sequences. You can check the related service provider with the naming pattern `<ProdCon_name>_CloudServer` in the **Manage Service Providers** app to see if it exists. If not, edit the corresponding Production Connector under the **Production Connectors** tab of the **Configure Production Connectivity** app and save it again. If the problem persists, open a support ticket at http://support.sap.com/ on component `MFG-DM-MA-PCM`.

Process Runtime Web Server	Web Server to Connect	Object Included	Note
Production Connector on Windows web server	(not required)	functions	[9]
		shop floor services (OPC UA methods)	[10]
		read/write indicators (built-in services)	[15], [2], [23]
		read/write global variables (built-in services)	[20], [23]
		array/map services (built-in services)	[28]
		publish MQTT message (built-in services)	[29]
		[Production Connector on Windows] automation sequences (subprocess)	[14]
	SAP Digital Manufacturing services type web server	cloud processes (subprocess)	[17], [13]
		SAP Digital Manufacturing cloud services	[1], [21], [11], [12]
		registered cloud processes	[5], [21], [11], [12]
	SAP Digital Manufacturing for edge computing services type web server	SAP Digital Manufacturing for edge computing services	[1], [22]
		edge processes (subprocess)	[17], [13]
		registered edge processes	[5], [22]

Process Runtime Web Server	Web Server to Connect	Object Included	Note
	user-defined web server (cloud)	third-party services (RESTful)	[6], [12], [7]
	user-defined web server (on-premise)		
Production Connector on edge web server	(not required)	functions	[9]
		shop floor services (OPC UA methods)	[10]
		[Production Connector on edge] read/write indicators (built-in services)	[15], [2], [23]
		read/write global variables (built-in services)	[20], [23]
		array/map services (built-in services)	[28]
		publish MQTT message (built-in services)	[29]
		[Production Connector on edge] automation sequences (subprocess)	[14]
	[automatically connected by system] SAP Digital Manufacturing for edge computing services type web server	edge processes (subprocess)	[25], [13]
		SAP Digital Manufacturing for edge computing services	[1], [25], [26], [12]
		registered edge processes	[5], [25], [26], [12]

Edge runtime environment

You should have installed SAP Digital Manufacturing for edge computing, deployed the **Production Process Orchestration** module, and performed certain configurations in the Cloud Connector before creating a production process in an edge runtime environment. For details, see Setup and Operations Guide for SAP Digital Manufacturing for edge computing, Modular Deployment, and Enable Deploying Edge Processes from Cloud to Edge.

Process Runtime Web Server	Web Server to Connect	Object Included	Note
SAP Digital Manufacturing for edge computing services type web server	(not required)	SAP Digital Manufacturing for edge computing services	[1]
		[Production connector on edge] read/write indicators (built-in services)	[15], [2], [27]
		[Production connector on edge] automation sequences (subprocess)	[15], [8], [25]

Process Runtime Web Server	Web Server to Connect	Object Included	Note
		read/write indicators flexibly (built-in services)	[18]
		read/write global variables (built-in services)	[20]
		array/map services (built-in services)	[28]
		edge processes (subprocess)	[24]
		registered edge processes	[5], [24]
	user-defined web server (cloud)	third-party services (RESTful, OData or SOAP)	[6], [16], [7]
	user-defined web server (on-premise)		

Note

1. API services provided by SAP Digital Manufacturing. For details, see Registering Services in the SaaS Tenant of SAP Digital Manufacturing.

   Some services are available to SAP Digital Manufacturing for edge computing services type web server. They can be checked in the **Manage Web Servers** app.

2. Read or write values from indicators on asset.

   There are two types of indicators available: 1) indicator mapped to a primitive data type tag; 2) indicator mapped to a component (primitive data type) of a structured data type tag. Note that only production processes and automation sequences of Production Connector runtime support the second indicator type. In such cases reading or writing indicator value only acts on the value of the component mapping to the indicator. Other component values under the same tag are not affected. See Mapping Indicators to Tags for details.

   You can check the service details in the **Manage Service Registry** app.

3. Read or write values from indicators locally. Without bothering to assign a shop floor system to the asset, you can save the value temporarily to a local indicator (write) and reuse (read) it later.

   You don't need to onboard the asset first before using indicators in the Production Process Designer. Both mapped and unmapped indicators are supported for reading or writing indicators locally.

   You can check the service details in the **Manage Service Registry** app.

4. Includes POD plugins originally available to be called in the Production Process Designer and custom POD plugins developed by users. For details, see Supported POD Plugins in Production Process Designer and Controls. A blue row highlight indicator is attached to custom POD plugins developed by users. There are two types of custom POD plugins:

   a. Type A: Custom POD plugins deployed on an external PaaS tenant. You need to register them in the **Manage Service Registry** app first. For details, see UI Extensions.

   b. Type B: Custom POD plugins deployed directly in POD Designer can be used right away without any registration procedures. For details, see Deploying in POD Designer.

   In case there are duplicated IDs or names when you have both type A and type B POD plugins, what the production process designer displays is based on the rules in the following table:

POD Plugin Type	Duplicated ID with Different Name	Duplicate ID with Same Name	Duplicated Name with Different ID
Type A and Type B	Type B	Type B	Type A + Type B **Note** Choose the icon to see the source and extension information for each type. The extension field shows the name of the UI extension in the service registry or the custom extension name in POD Designer.

5. Cloud/edge processes published from the Production Process Designer to the **Manage Service Registry** app.
6. Any services that you manually register with SAP Digital Manufacturing in the **Manage Service Registry** app. Not all RESTful, OData and SOAP services are supported. For details, see Registering Services in the SaaS Tenant of SAP Digital Manufacturing.
7. Connect the runtime web server with the web server you created to use the services. For details, see Manage Web Servers.
8. Check **Visible to Cloud/Edge Runtime** to make the automation sequence you created visible to the current runtime.
9. For details, see Predefined Functions in the Multiple Call Destination System.
10. These services are provided by external service providers that are created on the selected runtime Production Connector system. Besides, a client proxy must be configured for each service. For details, see Configure Shop Floor Services (OPC UA Methods). You need to select the client proxy when using the service or production process.
11. Connect the runtime web server with the SAP Digital Manufacturing services type web server or SAP Digital Manufacturing for edge computing services type web server. Then deploy services you want to use from the connected web server. For details, see Manage Web Servers and Adding and Deploying Services from a Connected Web Server.
12. A client proxy must be configured for each service or production process. For details, see Configure Client Proxy for Third-Party Service. You need to select the client proxy when using the service or production process.
13. Check **Visible to Production Connector / Plant Connectivity Runtime** to make the cloud/edge process you created visible to Production Connector runtime.
14. The main processes and subprocesses must be run by the same Production Connector system.
15. To use indicator services, create an automation sequence or a subscription, you should have configured the Production Connectivity Model with the shop floor system added first. For details, see About Production Connectivity Model.
16. For details, see Use Third-Party Service in Cloud/Edge Process.
17. Connect the runtime web server with the SAP Digital Manufacturing services type web server or SAP Digital Manufacturing for edge computing services type web server. This ensures that relevant processes are available for use in the left panel of the production process designer. For details, see Manage Web Servers and Edge Web Server.
18. Read or write values from indicators you decide through string constant values, local variables, parameters, or expressions. Designate the value of the asset name, reference path (reference name of the root structure + “/” + reference name of the structure), and reference name through variables in the value help for the system to dynamically identify the indicator. For details on asset, reference path (structure), and reference name, see related pages under Asset Model.

For example, for “indicator1” you created under a multi-level structure of “TopGroup1/SubGroup1” in the asset “DemoAsset1” in the **Manage Assets** app, the input value of this indicator can be:

- assetName: `"DemoAsset1"`
- referencePath: `"TopGroup1/SubGroup1"`
- referenceName: `"indicator1"`

If you have created multiple such indicators in different assets, the read/write indicators flexibly service provides you great flexibility to use all these indicators through dynamic parameters. In the field of **Value From** of each line, you can add a string constant value, local variable, parameter, or expression.

The service will only output “400” (with the Production Connector error code for reading/writing failures in the Production Connector) in its HTTP response when all reading/writing indicators fail. You can attach an error catch element to deal with this situation. In partial failure cases, the output will still be “200” for client validation failure. If there are reading/writing failures in the Production Connector, the Production Connector error code will be returned. You can process these successful and failed calls with the condition element.

You can check the service details in the **Manage Service Registry** app.

19. If a cloud process is both a subprocess created in the current production process design and a process published to service registry, you should use the subprocess one under the “Production Processes” folder in the left panel instead of the one registered in the service registry (under the “DMC_Cloud/Production Process” folder).
20. Read global variables: Read service for the specific global variable. You can only read one global variable each time.

    Write global variables: Write service for the specific global variable. You can write the input parameter value to the global variable and pass the output value of the service to the following steps. You can only write one global variable each time. For the input parameter of the Write Global Variables service, the expression editor is supported only when the global variable is of primitive data type (primitive array not included). There are some limitations for automation sequences. For details, see 3379404.

    See Global Variables for details.
21. For details, see Create a Destination for SAP Digital Manufacturing.
22. For the prerequisite of this use case, see Enable Production Connector to Invoke Edge Services.
23. Read indicators services are only available for Production Connector 2.1.0 or above. Read/write global variables services are only available for Production Connector 2.0.0 or above.
24. The main processes and subprocesses must be run in the runtime environment of the same edge web server.
25. The main processes and subprocesses must be run on the same edge device.
26. Deploy services you want to use from the connected web server. For details, see Adding and Deploying Services from a Connected Web Server.
27. The edge process and the shop floor system of the indicator must be run on the same plant.
28. Built-in services for array or map variables.

Service	Description	Input → Output	Example `array1: ["a", "b", "c"], map1: {"key1": "v1", "key2": "v2"}`
Get Array Length	Return the number of items in the given array.	array variable → number of items (integer)	`array1` → 3

Service	Description	Input → Output	Example `array1: ["a", "b", "c"], map1: {"key1": "v1", "key2": "v2"}`
Get Array Item	Find an item in an array based on its index.	array variable, index (integer) → array item	`array1, 2 → "c"`
Add Array Item	Append a new array item.	array variable, array item → new array	`array1, "d" → array1: ["a", "b", "c", "d"]`
Update Array Item	Update an item in an array for a given index and item.	array variable, index (integer), array item → updated array	`array1, 1, "g" → array1: ["a", "g", "c"]`
Remove Array Item	Remove an item from an array based on its index.	array variable, index (integer) → updated array	`array1, 2 → array1: ["a", "b"]`
Get Map Size	Return the number of items in the given map.	map variable → number of items (integer)	`map1 → 2`
Get Map Item	Find an item value in a map for a given item key.	map variable, item key (string) → item value	`map1, "key1" → "v1"`
Set Map Item	Add or update an item in a map for a given item key and item value.	map variable, item key (string), item value → new/updated map	`map1, "key3", "v3" → map1: {"key1": "v1", "key2": "v2", "key3": "v3"}`
Remove Map Item	Remove an item from a map for a given item key.	map variable, item key (string) → updated map	`map1, "key1" → map1: {"key2": "v2"}`

The expression editor is not supported in the above services' input parameters.

Services for map variables should be used after the Read Global Variables service. For details on array and map data type, see Data Types and Data Type Conversion.

There are some limitations for automation sequences. For details, see 3379404.

29. Publish an MQTT message to a message broker. There are some prerequisites:

    - Create the message broker in the **Manage Message Brokers** app. Connect a Production Connector (version 2.0.0 or higher) to the message broker you created. For details, see Production Connectors.
    - Under the **Schemas** tab of the **Manage Service Registry** app, configure a schema that will be used for the message to subscribe on the message brokers. Note that this is needed only if your message payload is a complex data type. In case the message payload is just a single string or numbers, schema definition is not a prerequisite. For details, see Create a Schema.

    You need to select the message broker and input the topic to which you will publish your message. You can use free text, parameter, variable or expression when defining the topic. Make sure you have Production Connector on Windows or Production Connector on edge, both with version 2.6.0 or higher, when using this feature. Refer to the MQTT official documentation for topic name rules. Only message brokers connected to the same Production Connector web server as the current automation sequence are available to select. Select the message payload data type first and then select the message payload schema if needed and the QoS level (default is 2 - Exactly Once) of the message to be published. Assign the values for each property in the message payload. The values support constant, or structure, which is the output of previous services.

Configure Service Library

Services that are available for use depends on the runtime environment of the particular process. You can always configure the service library to hide and show services to facilitate your design work.

Procedure

1. Open a production process in the editor.
2. On the left panel, under **Services and Processes**, choose **Select Services**.
3. In the **Select Services** window, select to show and deselect to hide a service. Choose **Refresh** to reload the service list.

   > **Note**
   > Instead of selecting the top group, you need to select the immediate parent group of the service for it to be available in the left panel.

4. Save your changes.

Controls

Controls are visualizations of different BPMN event elements. Some controls are applicable only to the cloud process; some controls have different usages in the different categories of production processes.

Control	Cloud Process / Automation Sequence / Edge Process
 Start	Cloud Process, Automation Sequence, Edge
 End	Cloud Process, Automation Sequence, Edge
 Condition	Cloud Process, Automation Sequence, Edge
 Parallel	Cloud Process, Edge
 Error Catch	Cloud Process, Edge
 Error End	Cloud Process, Edge
	Cloud Process, Edge

Control	Cloud Process / Automation Sequence / Edge Process
Wait	
 Script Task	Cloud Process, Edge
 POD Message	Cloud Process
 Annotation	Cloud Process, Automation Sequence, Edge

Start

A start element represents the first step in a process. Each process has only one start element.

Use the start element to define process input parameters.

> **Note**
> Start element supports local variables only.

End

An end element represents the last step in a process. Each process has only one end element.

Use the end element to define process output parameters.

Condition

Use a condition element to create multiple flows in a process but only one flow is executed under a given condition. Ensure there is only one flow before the condition element.

- For a cloud/edge process, you should create two or more flows after the condition element.

  For each flow, define an evaluation expression. The expressions are evaluated **sequentially** following the given order. In other words, an expression is only evaluated when it is the first expression or the previous expressions are all evaluated to `false`.

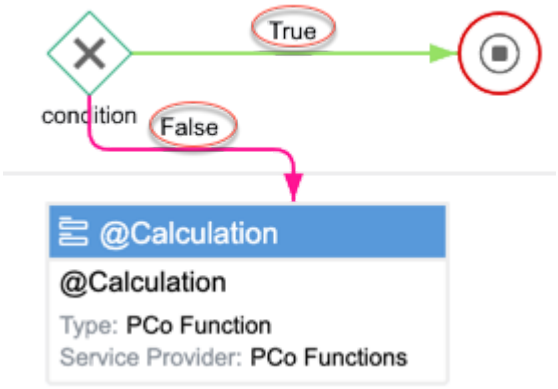
  When an expression is evaluated and the result is `true`, the corresponding flow is executed. The other flows are all ignored.

  > **Note**
  > You can drag and drop the evaluation expressions to adjust the evaluation order.

  > **→ Recommendation**
  > Define the last evaluation expression as the ELSE clause so as to ensure that there is always one flow to execute.

- For an automation sequence, you should create only two flows after the condition element.

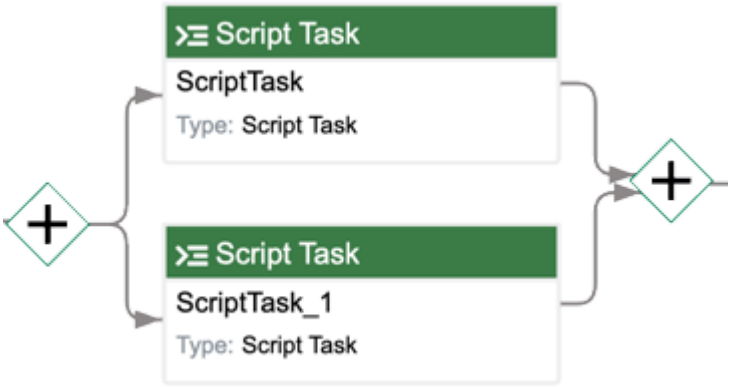
  Adjust the evaluation order of the two flows by double-clicking either `True` or `False` on the lines. And then change `True` to `False`, or vice versa.



Parallel

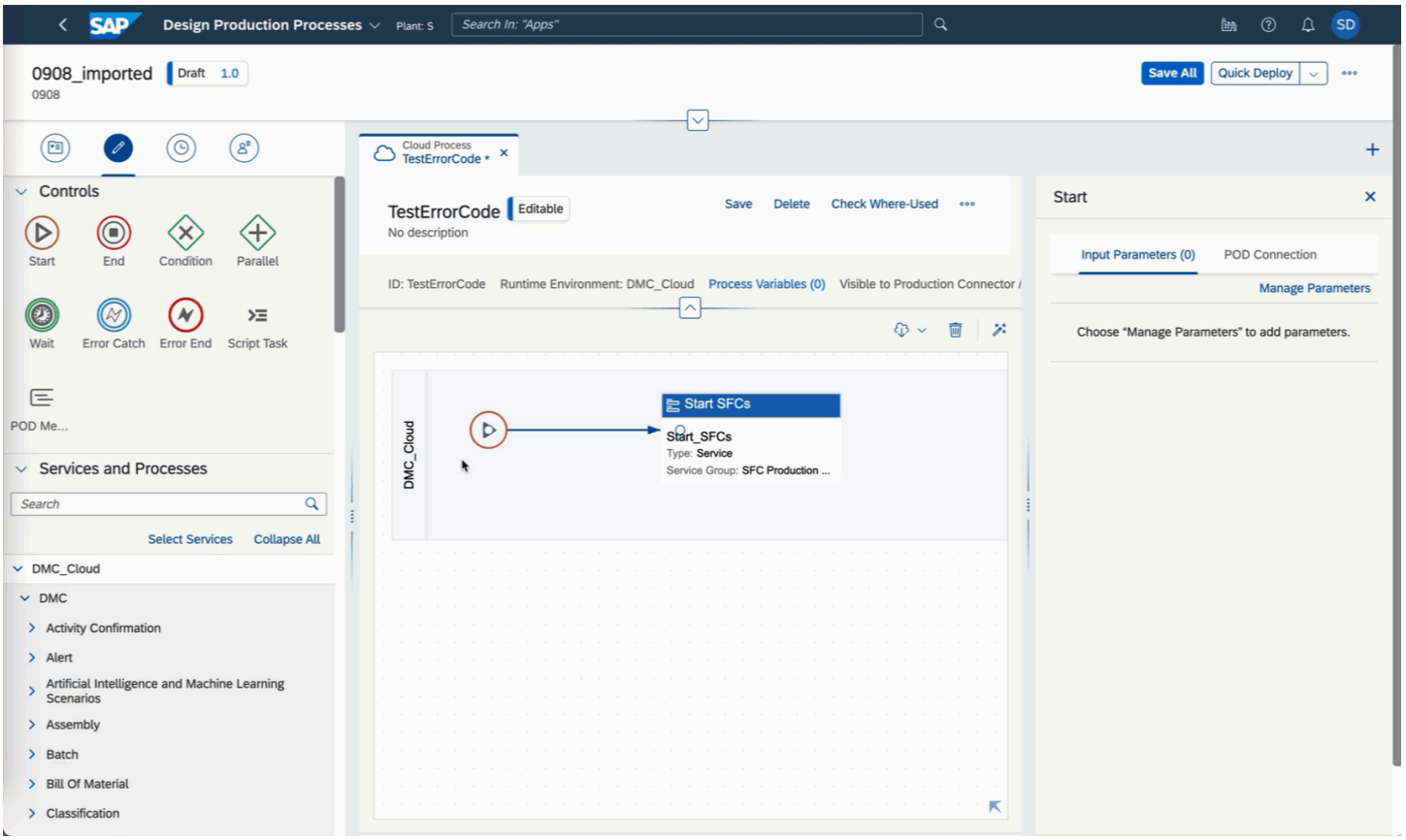
Use a parallel element to create multiple flows that are executed simultaneously to save the overall execution time of production process.

You should use the parallel elements in pairs. All parallel branches should be started from one parallel element and joined together to another common parallel element at the end. Ensure there is only one flow before the first parallel element and after the second parallel element.



Error Catch

An error catch element catches errors codes thrown by a service/process on which it is defined. Use the error catch element to create alternative flows that handle possible error states the service/process might run into.



→ Tip

1. Drag the error catch element from the left panel.
2. Put the error catch element on the border area of the Start SFCs service element on the canvas, with the service highlighted. Release the error catch element and you will see it is attached to the service element.
3. Select the service error codes you want to catch on the right panel of the error catch settings.
4. Drag and drop the error end element on the canvas. Create a flow from the error catch to the error end element. On the right panel of the error end settings, select the error code you have defined.

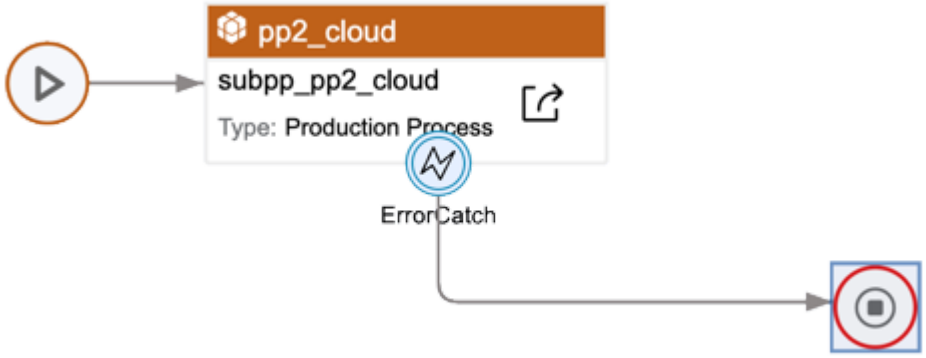
Object	Error code caught by the error catch element
SAP Digital Manufacturing cloud services, indicator services	HTTP status codes
third-party services	HTTP status codes
cloud/edge processes (subprocess)	error codes defined in the Manage Error Codes settings
automation sequences (subprocess)	HTTP status codes
registered cloud/edge processes	HTTP status codes

The HTTP status codes cover standard response codes that you can refer to Standard Response Codes of Service and Production Process.

The error code caught by the error catch element can be used as a local variable afterwards in these types of flow tasks in the alternative flow:

- Service: input parameter
- Script task element: input parameter
- Subprocess: input parameter
- End element: output parameter

ℹ Note



The execution result status for a process catching an error and successfully running to the end is **Completed** in the **Monitor Production Processes** app. See Monitor and Track Production Processes for details.

- Condition element: condition expression

For example, when editing an expression in the condition element, you can input directly `errorCode` and choose the error code from the dropdown list.

- If you want to catch errors (HTTP status codes) thrown by an SAP Digital Manufacturing cloud service, indicator service, third-party service, registered cloud/edge process, or automation sequence (subprocess), follow the steps below:

    1. Attach an error catch element to the service or subprocess and define which HTTP status code (401, for example) this error catch element catches.
    2. In the alternative flow leading from the error catch element, add a condition element and define one of the evaluation expressions with the error code variable in the expression editor as `'errorCode' == "HTTP401"`. To do so,
        a. Choose **Functions and Variables** > **Variables** > **'errorCode'**.
        b. Input " == ".
        c. Choose **Functions and Variables** > **References**.
        d. Choose **HTTP401** from the list error codes you have defined in Step 1.

- If you want to catch errors (error codes defined in the **Manage Error Codes** settings) thrown by a cloud/edge process (subprocess), follow the steps below:

    1. Define error codes in the **Manage Error Codes** settings. For example, add “StartSfcFailed” and “Start SFC failed.” as the **Key** and **Message**.

2. In the subprocess, add one or more error end elements and choose in the right panel from the list of error codes you have defined in Step 1.

3. In the parent process, attach an error catch element to the subprocess and define which error codes (StartSfcFailed, for example) this error catch element catches.

4. In the alternative flow leading from the error catch element, add a condition element and define one of the evaluation expressions with the error code variable in the expression editor as `'errorCode' == "StartSfcFailed"`. To do so,

    a. Choose **Functions and Variables** > **Variables** > **'errorCode'**.

    b. Input `" == "`.

    c. Choose **Functions and Variables** > **References**.

    d. Choose **StartSfcFailed** from the list error codes you have defined in Step 3.

For more information, see Define Error Codes for Error End Events.

You can also attach an error catch element to POD plugin, or POD message with **Require Confirmation** checked. Select from the following error codes:

- **INTERNAL**: The POD has encountered a fatal internal error that is technical.
- **POD_CONFIGURATION**: A POD configuration problem is preventing the user from proceeding.

Error End

Use the error end element to display an error message when a process runs into an error.

When a process runs into an error end, the process ends and throws an error message.

You have the option to re-launch asynchronously-run cloud processes terminated with error in a central place later. To do so, switch on the **Show Process in Recovery List** toggle. Choose from the following termination modes so that the production process will appear in the recovery list based on different reasons:

Termination Mode:

- **Failure**: A condition on input parameters that became true (e.g. parameter out of range)

    > **Note**
    > You can specify the input parameters causing the failure in the dropdown list.

- **In Queue**: Due to an external context that makes the process waiting (e.g. resource/equipment blocked)

You can see these production process instances in the **Recover Production Processes** app. See Recover Production Processes for further actions.

Wait

Use a wait element as a placeholder to delay the following process step from execution for a specified duration. For each wait element, provide a name and input a constant value, an expression, or a local variable to specify the wait time from 1 to 3600 seconds in the attribute panel.

ℹ Note

The wait element is treated as a usual step and cannot be used at the beginning or end of the whole process. When the process is executing the wait step, the process status in the **Monitor Production Processes** app is **Running**.

To avoid timeout issues, if the wait time is set to over 30 seconds, you are recommended to run the process asynchronously.

Beware that a system delay within 10 seconds might add to the actual wait time in the runtime environment.

Script Task

A script task is an activity/service defined by users in JavaScript to enhance some simple logics in production process designer for cloud/edge runtime. When a production process execution arrives at the script task, the corresponding script is executed.

For each script task, provide a name and description (optional) in the attribute panel. Define input parameters and output parameters. Enter your script code in the code editor. Choose **Insert Parameter** to quickly insert predefined parameter names into the code. Choose *(format)* to apply automatic indentation.

ℹ Note

The JavaScript of script task follows org.mozilla.rhino-1.7.13, with partial support of ECMAScript 6. There are reserved keywords that cannot be used as local variable names assigned to parameters of script task. For details, see 3379404 .

Name of the local variable in script and input parameter or local variable should be different. Otherwise, it may lead to execution errors in runtime.

The color theme of the script task code editor might not work well under these themes: SAP Belize, SAP Belize Deep, and SAP High Contrast Black (Belize).

Only the following types or objects are supported in script task output parameter.

Supported Type / Object	Example
Null Type	`var a = null`
Boolean Type	`var a = true` `var b = false`
String Type	`var a = "abc"`
Number Type	`var a = 1` `var a = 1.1`
JSON Object	`var a = {"FullName": "Full Name"}`
Array Object	`var a = ["a","b","c"]` `var b = [1,2,3]`

You can use any supported types or objects in the script task code internally, but need to convert them to the above types or objects before sending to script task output parameter. For example, before sending to script task output parameter, use `var a = (new Date()).toISOString()` to get current date time and convert to string.

Choose **Test Run** to do code validation before deployment.

ℹ Note

Test run button is only enabled for the creator of the production process.

You can also use codes, for example, `var localDateTimeString = fn.localDateTime("2022-02-15T04:26:45.123Z","Europe/Berlin")`, to convert to your preferred execution timezone. Note that this feature is only applicable to deployed production processes. Test run is not supported.

ℹ Note

The execution of script tasks is subject to resource limits, for example, with respect to processing time or memory usage. For the specific limits that apply to script tasks, see the related section in 3379404 for details.

POD Plugin and POD Message

You can call POD plugin in cloud production process. The production process will call the POD to take action when the POD plugin service is executed. For a full list of POD plugins originally available to be called in the Production Process Designer and their descriptions, see Supported POD Plugins in Production Process Designer. You can also use custom POD plugins deployed on an external PaaS tenant or deployed directly in POD Designer. For details, see UI Extensions and Deploying in POD Designer.

Use POD Message in cloud production process to define message shown in a specified type of POD. Process with steps asking for user's action from POD will wait upon user's further action in POD to proceed with its following steps when it is triggered. You can also view its status as “Running (waiting)” (counted in “Running” KPI carousel) in the **Monitor Production Processes** app. For details, see Monitor and Track Production Processes.

1. Configure settings for POD plugin/message in the **POD Connection** tab of **Start** control.

   ### ℹ Note

   This step is a prerequisite for using POD plugin/message in the process later.

   At the same time, you need to do some configurations on the POD side:

   - Select **Production Process** in the event section of POD Notifications in the **POD Designer** app. For details, see POD Notification Options.
   - Check the connectivity status to make sure connection is established between POD and process.
   - Use the Process Overview plugin on the POD to check the execution status of the process.

2. Choose **Add**.
3. Select a POD type (Work Center, Operation Activity, Order) in which the POD plugin/message takes effect.
4. Designate corresponding metadata in **Conditions to Select POD** so that the production process knows which POD to call when the POD plugin/message is executed.

   ### ℹ Note

   The metadata that you designate in **Conditions to Select POD** must match the **Subscription** options that you select when configuring POD Notifications in the POD Designer. For more information, see POD Notification Options.

   For example, when configuring **POD Notifications** in the **POD Designer**, you select **Work Center** and **Resource** in the **Subscription** section and ensure that **Production Process** is one of the events selected in the **Event** section. Then, when configuring **Conditions to Select for POD** in the **Design Production Processes** app, you ensure that the production

process sends processes to the POD that are listening for both the same **Work Center** and **Resource** by designating the same **Work Center** and **Resource** that you selected when configuring **POD Notifications** in the **POD Designer**.

> **Note**
>
> Both input parameters of the **Start** control or constant values in the value helper are supported.

5. Drag and drop a POD plugin in the service library to the canvas.
6. For each plugin, give it a name and select an action type in the attribute panel. There are two action types:
   - **Launch and Wait**: Execute the POD plugin and wait for input from operator to continue. In the POD, choose **Continue** to finish POD action and continue to next step of process.
   - **Launch and Continue**: Execute the POD plugin and continue to the next step without further input.

   When POD plugin is executed, selected plugin will appear automatically on the selected POD.

> **Note**
>
> It is recommended to use asynchronous call if POD plugins are run with "Launch and Wait", otherwise it may cause timeout.

7. Edit more attributes in the attribute panel for POD plugins that support input and output parameter configuration.
8. You can also drag and drop a POD Message control to the canvas.
9. Give the POD message a name and input a message in the attribute panel.

> **Note**
>
> Choose **Insert** to add the input parameter of the start element, local variable, or output parameter of the previous element in the POD message. Only primitive data type and primary node of the structured date type (the node with primitive data type that has not any sub-node, for example: `person[0].name`) is supported. Apply escape character (`\" \"`) if you want to use double quotes (`""`) in your message. You can also add your own expression. Some reserved words are not allowed. For details, see 3379404.
>
> Message translation is not supported when you change the language setting.

10. Check the **Require Confirmation** checkbox to ask for operator's confirmation in a dialog in POD. Otherwise, the message will appear as message toast on the selected POD and the process is continued without confirmation from POD.

> **Note**
>
> The following scenarios occur when there are multiple production processes in your production process design:
>
> When you use the **Asynchronous** call type, the POD doesn't wait for the **POD Message** to be displayed and proceeds to the next production process.
>
> When you use the **Synchronous** call type, the POD waits for the **POD Message** to be displayed and then proceeds to the next production process when the confirmation is provided through the **POD Message** popup.

Annotation

An annotation element helps you add descriptions to individual production process steps or the entire production process.

Use Shift + Enter to wrap the line and Backspace to delete the annotation. Support a maximal length of 256 characters.

Define Error Codes for Error End Events

In a production process design, you can define error codes and assign them to error end events in various cloud processes. Each error end event can throw errors defined by one particular error code.

Prerequisites

You have the role of `Automation_Engineer` or `Production_Engineer`.

Procedure

1. Open a production process design.
2. Choose **Manage Error Codes** in the upper right corner of the header section.
3. Create one pair of key and message for each error code.

ℹ Note

Some reserved words are not allowed. For details, see 3379404.

Example

- **Key**: `sfcStartInsufficientQty`
- **Message**:
  `SFC could not be started because there was no sufficient quantity for the material`

  Note that local variables are not supported for the message. For example, `SFC 'sfcV' not started` is **not** supported.

Related Information

Controls

Supported POD Plugins in Production Process Designer

This is a full list of POD plugins originally available to be called in the Production Process Designer.

ℹ Note

You can also use custom POD plugins deployed on an external PaaS tenant or deployed directly to POD Designer. For details, see UI Extensions and Deploying in POD Designer.

POD Plugin	Description	Reference Link
Activity Confirmation	Report production activities at the phase level of a process order or at the operation level of a production order.	Activity Confirmation
Assemble Components	Record assembly work on the shop floor and enter assembly data values for the material at an assembly point for one SFC.	Assemble Components

POD Plugin	Description	Reference Link
Change Equipment Status	Change the status of a single resource in a work center to indicate that the resource is unavailable to be used in manufacturing at a specific time.	Resource Status with Reason Code
Complete	Complete the work on selected SFCs at your operation activity and resource.	Complete
Component List	View the list of all the components defined in the BOM which then will be filtered to the selected operation activities for the selected SFC in the POD.	Component List
Data Collection	Collect production process-related data, entered against SFCs in the POD.	Data Collection
Electronic Signature	A verification or approval step in your manufacturing process.	Electronic Signature
Goods Receipt	Trigger goods issue and goods receipt, post and view goods receipt quantity.	Goods Receipt
Guided Steps	Show the plugins required to execute an operation activity or phase to the Operator.	Guided Steps - Production Process
Labor Off	Manually stop tracking the time you're working on an SFC at an operation activity.	Labor Off
Labor On	Start tracking the time you work on an SFC at an operation activity.	Labor On
Logistics Order Creation	Enable automatic creation or update of logistics orders by assigning destination locations.	Logistics Order Creation
Material Consumption	View a list of materials required for consumption with target and total consumed quantity at phase level.	Material Consumption
Nonconformance Data Entry	View information on the logged NC codes and change states of the NC codes.	Nonconformance Data Entry
Packing List	Pack and unpack SFCs and packing units in the POD.	Packing List
Quality Inspection Characteristic List	Display the inspection characteristics for an operation activity.	Quality Inspection Characteristic List
Quality Inspection Results	Record the inspection results for an operation activity	Quality Inspection Results
Quantity Confirmation	Report and display the yield and scrap quantities posted at an operation/phase of an order.	Quantity Confirmation
SFC Merge	Merge several SFCs that are in the New or In Queue status and belong to the same material and material version, BOM, routing, operation activity, and order into	SFC Merge

POD Plugin	Description	Reference Link
	one SFC that is in the New or In Queue status.	
SFC Relabel	Rename an SFC in the POD.	SFC Relabel
SFC Serialize	Ungroup a product by splitting an SFC into individual pieces with its own unique SFC names in the POD.	SFC Serialize
SFC Split	Split any quantity of one SFC into a new SFC.	SFC Split
Tool Logging	Manually log tools at an operation activity of an SFC.	Tool Logging
Work Instruction Viewer	View the list of available work instructions that you have selected in the POD.	Work Instruction Viewer

Local Variables

Variables defined locally within a process are used to map parameters between elements within a process. They are not inherited by subprocesses or detectable by the parent process.

ℹ Note

Apart from local variables created manually, you can also use error codes as local variables. For more information, see Error Catch.

Create a local variable for the following purposes:

- Assign a process input parameter to a local variable
- Assign an output parameter to a local variable

The local variables can be used to do the following:

- Define expressions for input parameters
- Define Boolean expressions for conditions that evaluate to either `true` or `false`
- Define values in arrays

The local variable is visible in a single instance of a single process. The variable name must be unique in the scope of a production process definition.

ℹ Note

The local variable name only supports limited western characters (A–Z, a–z, 0–9, _). The default value should be no longer than 5000 characters. Some reserved words are not allowed. For details, see 3379404.

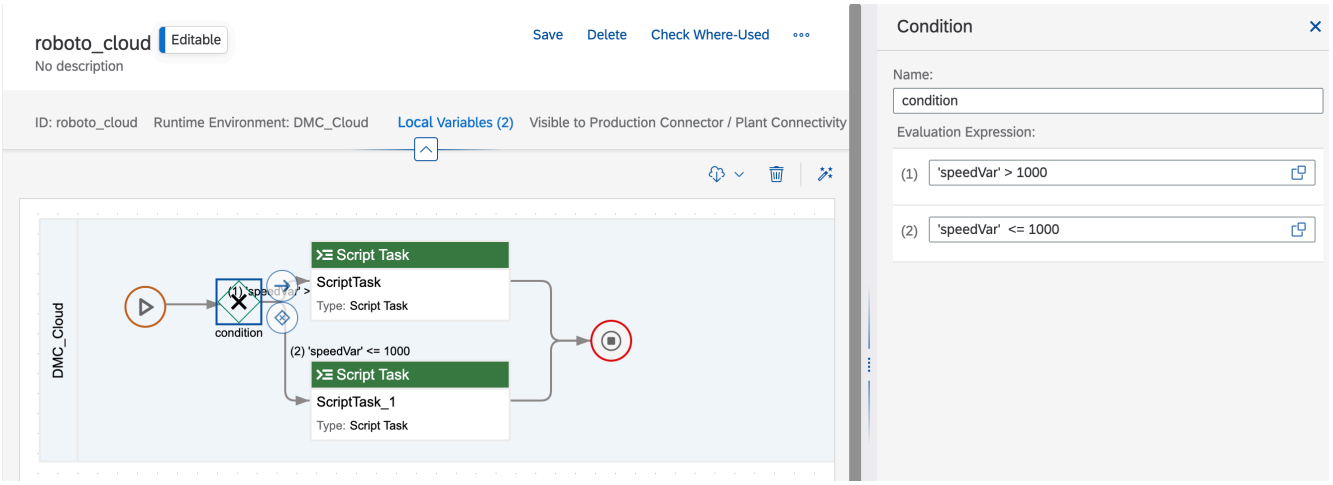
See Data Types and Data Type Conversion for supported local variable data type.

Example

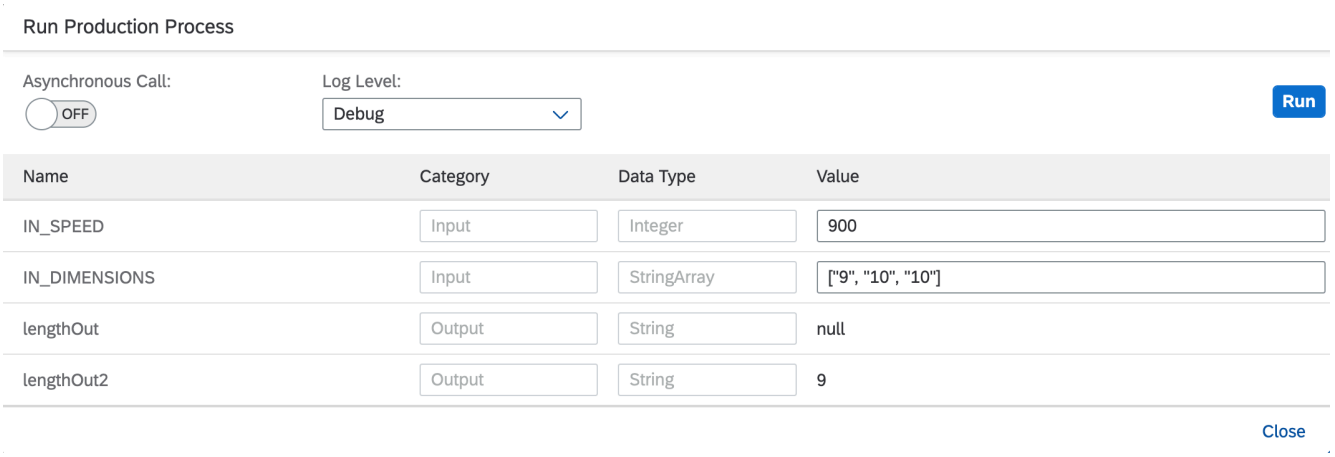
In this example, local variables are used to facilitate the mapping of input parameters to different process elements within a production process. The production process aims to perform certain operations based on the values of the input parameters and produce output parameters accordingly.

The production process involves two input parameters: `IN_SPEED` (an integer) and `IN_DIMENSIONS` (a string array). These input parameters are mapped to local variables `speedVar` and `dimensionsVar`, respectively, which can be used throughout the process. This allows for easier manipulation and management of the input data within the process workflow.

1. In the process header section, choose **Local Variables**. Create an integer local variable `speedVar` and a string array local variable `dimensionsVar`.
2. Create two process input parameters on the start element:
   - Integer parameter `IN_SPEED`, assigned to local variable `speedVar`
   - String array parameter `IN_DIMENSIONS`, assigned to local variable `dimensionsVar`
3. Add two script tasks with the following settings:
   - Flow A: Define both the input and output parameters as `length` (data type: string) with the input parameter value as `'dimensionsVar[0]'`. Define the script as `$output.length = $input.length`.
   - Flow B: Define both the input and output parameters as `length2` (data type: string) with the input parameter value as `'dimensionsVar[0]'`. Define the script as `$output.length2 = $input.length2`.
4. Add a condition element and define the conditions as follows:
   - If `'speedVar' > 1000`, flow A is executed.
   - If `'speedVar' =< 1000`, flow B is executed.



5. Create two process output parameters on the end element:
   - String parameter `lengthOut`, assigned to parameter `'ScriptTask#length'`.
   - String parameter `lengthOut2`, assigned to parameter `'ScriptTask#length2'`.
6. At runtime, the input values for `IN_SPEED` and `IN_DIMENSIONS` are `900` and `["9", "10", "10"]`, respectively.
7. Flow B is executed.
8. The input parameter `length2` in step 3 receives "9" as its value.



Global Variables

Global variables are used to read and write data across all instances of a single process or all processes of the same runtime system in a single production process design.

Create a global variable for the following purposes:

- Access and reuse the data in other process instances, processes, or designs

  > **Note**
  >
  > Global variable supports a maximum memory of 2,097,152 bytes. Automatic retry and error catch are not supported. The global variable name only supports limited western characters (A–Z, a–z, 0–9, _). Some reserved words are not allowed. For details, see 3379404.

- Define the data to be buffered locally

  > **Note**
  >
  > You can define map data type as a global variable to buffer data locally and then manipulate this data with built-in map services provided by the Production Process Designer. For details, see Data Types and Data Type Conversion and Available Services and Subprocesses.

You can create global variables that are of different visibility scopes:

- **Global Variable (Process-wide)**: The global variable is visible in all instances of a single process. The global variable name must be unique in the scope of a production process definition.
- **Global Variable (Design-wide)**: The global variable is visible in all instances of all processes of the same runtime system in a single production process design. The global variable name must be unique in the scope of the production process design.

To create a process-wide global variable, do the following:

1. In the design header section, choose the number under **Global Variable**.
2. Under the **Process-wide** tab, select from the dropdown list the production process where you want the global variable to be visible.
3. Choose **Edit** > **Create**.
4. Enter the name, description, and data type.

Note

See Data Types and Data Type Conversion for supported global variable data type. Select a schema if you choose `Structure` or `StructureArray` as the data type. You can change the visibility scope of a process-wide global variable under the **Visibility Scope** column.

5. Choose **Save**.

To create a design-wide global variable, do the following:

1. In the design header section, choose the number under **Global Variable**.
2. Under the **Design-wide** tab, choose **Edit** > **Create**.
3. Enter the name, description, and data type.

Note

See Data Types and Data Type Conversion for supported global variable data type. Select a schema if you choose `Structure` or `StructureArray` as the data type. Select the runtime web server under the **Runtime Environment** column. The global variable is only visible inside one runtime where it is defined. To use a global variable in the production process, ensure they have the same runtime web server.

4. Choose **Save**.

Note

When you edit a global variable that has been used in other places, the system will detect it automatically and you can check in the warning message box where this global variable is used once you save your edit. You can continue with your update or cancel. Alternatively, you can also choose **Dependencies** under the **Where-Used** column in the global variable list page to check where the global variable has been used.

If you change the global variable data type, the global variable in the runtime will be deleted and recreated after deployment.

After you update a global variable, the read/write global variables service will get the **Outdated (compatible)** or **Outdated (incompatible)** status. You will see a □ *(warning)* if the read/write global variables service has a compatible change. Choose **Synchronize** in the message strip or the □ *(synchronize)* in canvas to refresh it. If the change is incompatible, you will see an □ *(error)*. Choose □ *(synchronize)* to refresh it and reassign the values. Redeploy the production process design to make the update effective.

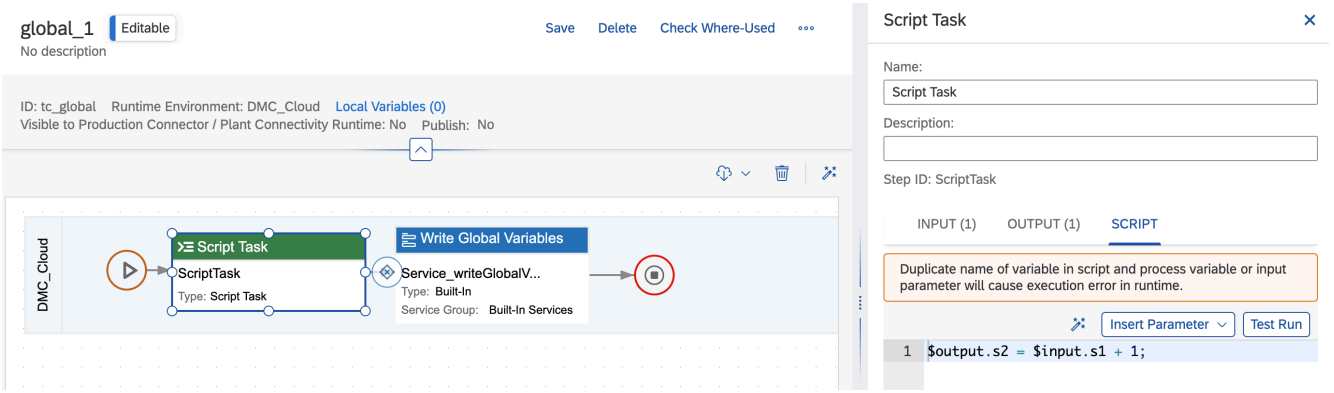
When you delete a global variable that has been used in other places, the system will detect it automatically and you can check in the warning message box where this global variable is used once you perform the delete action. You can continue with your deletion or cancel. If deletion is continued in this case, an □ *(error)* will be displayed next to the read/write global variables service in the related production process and the service can no longer be used.

Example

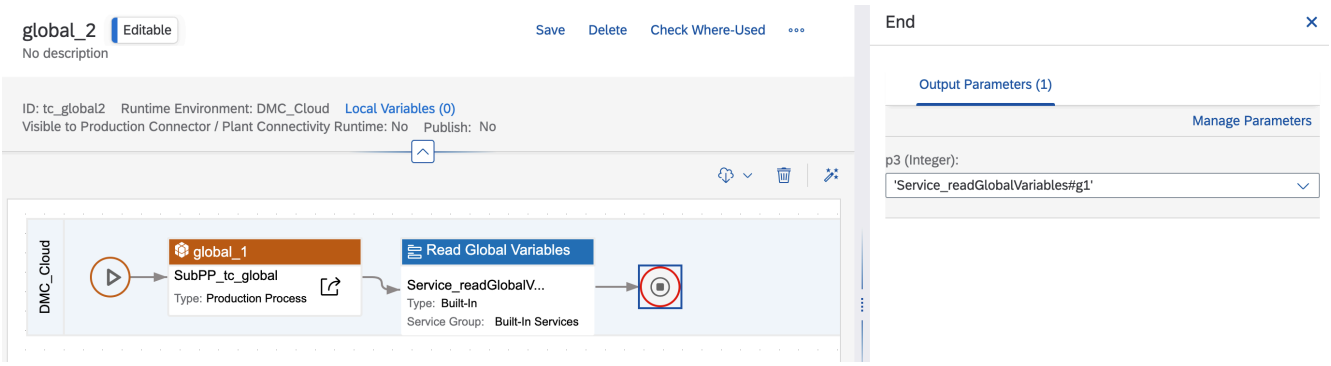
In this example, the global variable is used to share data between two separate cloud processes, `global_1` and `global_2`, within the same production process design. This production process aims to demonstrate how global variables can be utilized to facilitate communication and data transfer across different processes or instances of the same runtime system.

By using a global variable, the value computed in one process (`global_1`) can be accessed and utilized by another process (`global_2`), enabling seamless data sharing and integration within the overall production process. This approach can be particularly useful in scenarios where multiple processes need to collaborate or exchange information to achieve a common goal or complete a complex task.

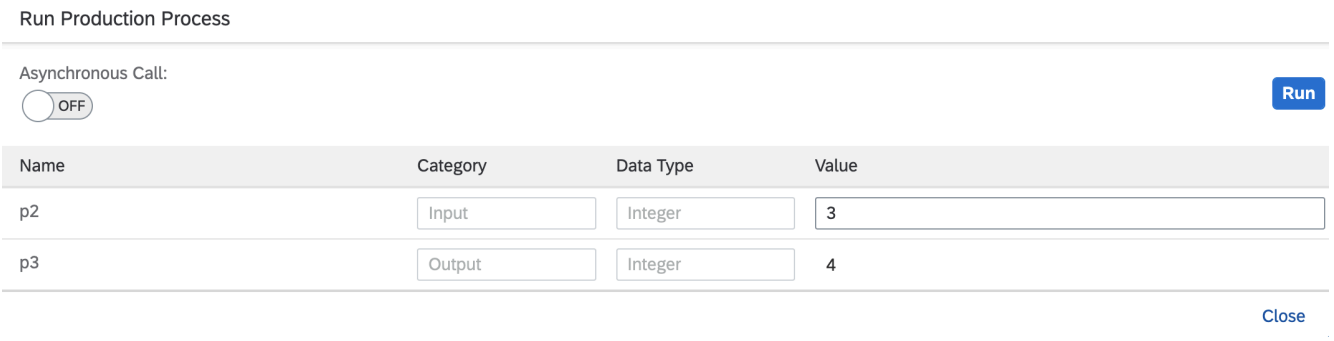
1. In the design header section, choose the number under Global Variable. In the Design-wide tab, create an integer global variable `g1` with the `DMC_Cloud` web server.
2. Create a cloud process `global_1`.
3. Create an integer input parameter `p1` on the start element.
4. Add a script task with the following settings:
    a. Define `s1` and `s2` (data type: integer) as input and output parameters separately.
    b. Define the script as `$output.s2 = $input.s1 + 1`.
5. Add Write Global Variables with the following settings:
    a. Add global variable `g1` as the input parameter.
    b. Choose `'ScriptTask#s2'` as its value.



6. Create another cloud process `global_2`.
7. Create an integer parameter `p2` on the start element.
8. Add `global_1` as the subprocess and choose `p2` as the value of its input parameter `p1`.
9. Add Read Global Variables and add the global variable `g1` as the output parameter.
10. Create an integer output parameter `p3` on the end element and choose `'Service_readGlobalVariables#g1'` as its value.



11. At runtime, the input value for `p2` is `3`.
12. The output parameter `p3` displays “4” as its value.



Data Types and Data Type Conversion

The Production Process Designer supports the following data types as local variable, global variable, and parameter values.

Primitive Data Type

A set of basic data types, including `String`, `Boolean`, `Integer`, `Short`, `Long`, `Float`, `Double`, `Byte`, and `DateTime`.

ℹ Note

`UInteger`, `UShort`, `UByte`, `Char` are also supported in an automation sequence and subscription.

String: To define a string parameter value, enclose the value with double quotation marks. For example, `"name"`.

Boolean: Only `true` and `false` (case-sensitive) are supported for Boolean values.

Primitive Array

An array of primitive data types, including `StringArray`, `BooleanArray`, `IntegerArray`, `ShortArray`, `LongArray`, `FloatArray`, `DoubleArray`, `ByteArray`, and `DateTimeArray`.

ℹ Note

`UIntegerArray`, `UShortArray`, `UByteArray`, `CharArray` are also supported in an automation sequence and subscription.

To define an array parameter with a fixed array value instead of assigning a local variable directly to it, enclose the value using square brackets. And if you want to use a value in an array local variable, define it following this pattern: `'variable[index]'`.

Example

Define a string local variable as `sVar` and a string array local variable as `arrVar`. And then define values for the following parameters:

- A string array parameter: `['sVar', "red", "green"]`
- A string array parameter: `['arrVar[0]']`
- A string parameter: `'arrVar[0]'`

Complex Data Type

A composite of primitive, primitive array, or other complex data types, including `Structure` and `StructureArray`.

Structure Data Type

An object that is a structure of property-value pairs. Structure data types are defined as schema in the **Manage Service Registry** app's schema library. See Schemas for details.

Structure Array

An array of structure data types.

Example

```
sfcs: [{material: "M12", shopOrder: "SO23", operation: "O34"}, {material: "M13", shopOrder:
```

Note

To designate structure data type (schema defined in service registry) or structure array to global variable, local variable, or input/output parameter value of cloud/edge production process and automation sequence, script task, and service, choose `Structure` or `StructureArray` as data type and then select the schema during creation. For details on adding parameter or variable values, see Add Value for Parameter or Local Variable.

Map

A data structure that stores a collection of key-value pairs, where each key is unique and associated with a single value, including `StringMap`, `BooleanMap`, `IntegerMap`, `ShortMap`, `LongMap`, `FloatMap`, `DoubleMap`, `ByteMap`, `DateTimeMap`, and `StructureMap`. `UIntegerMap`, `UShortMap`, `UByteMap`, `CharMap` are also supported in an automation sequence.

Example

```
sfcBuffer: {"SFC-123": {material: "M12", shopOrder: "SO23", operation: "O34"}, "SFC-124": {m
```

The map data structure can be used for buffering data locally. Instead of retrieving mission-critical business information via REST service calls, accessing locally buffered data greatly reduces latency. This is especially useful when such information is needed quickly during production processes.

Note

Map is supported only in global variables. The system provides a series of built-in map services to help you manipulate buffered data defined in global variables. See Available Services and Subprocesses for details.

Data Type Conversion

The following table describes the convertibility between different data types.

- A: Type can be cast or converted without any errors or precision loss.
- B: Type can be converted but sign loss may occur during conversion for negative values.
- C: Type can be converted but precision loss may occur during conversion.
- D: Type can be converted but it may overflow during conversion.

- E: Type can be converted but it depends on the validity of the input string.
- F: Type can't be converted.

Convert From \ Convert To	Boolean	UByte	Byte	Char	UShort	Short	UInteger	Integer	Single (Float)	ULong	Long	Double	DateTime
Boolean	A	A	A	A	A	A	A	A	A	A	A	A	F
UByte	C	A	A	A	A	A	A	A	A	A	A	A	F
Byte	C	B	A	B	B	A	B	A	A	B	A	A	F
Char	C	A	A	A	C	A	B	A	A	B	A	A	F
UShort	C	A	A	A	A	A	A	A	A	A	A	A	F
Short	D	D	D	D	D	A	B	A	A	B	A	A	F
UInteger	D	D	D	D	D	C	A	A	A	A	A	A	F
Integer	D	D	D	D	D	D	D	A	A	B	A	A	F
Single (Float)	D	D	D	D	D	D	D	C	A	C	C	A	F
ULong	D	D	D	D	D	D	D	D	C	A	A	A	F
Long	D	D	D	D	D	D	D	D	D	D	A	D	F
Double	D	D	D	D	D	D	D	D	D	D	C	A	F
DateTime	F	F	F	F	F	F	F	F	F	F	F	F	A
String	E	E	E	E	E	E	E	E	E	E	E	E	E

ℹ Note

Arrays and maps follow the same rules. For example, a byte array can also be converted to a string array but cannot be converted to a date time array.

Supported Operators in Expressions

You can use expressions to define parameters and conditions. Refer to the following table for the logical and mathematical operators that are supported in the expressions.

Operator	Description
(	Left parenthesis
)	Right parenthesis
>	Greater than
<	Less than

Operator	Description
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to
&&	Logical conjunction (And)
||	Logical disjunction (Or) This operator is used to enter an 'or' condition. **Example** Create this expression that evaluates to `true` or `false`: `'TAG_TEMP'<57 ||'TAG_TEMP'>63`.
+	Plus
–	Minus
*	Multiply
/	Divide
%	Modulo operation
^	Power **Note** This is supported for automation sequences only.
&	Concatenation You can use this operator to concatenate two string values. **Example** `'Temperature' & " degrees Celsius"` `Temperature` is the name of an integer local variable, enclosed in single quotes. The string value that is to be appended to the local variable value is enclosed in double quotes. At the runtime, if the value for `Temperature` is `44`, it is first converted into a string value and then combined with the string value as `44 degrees Celsius`. **Note** For the cloud runtime environment, you can also use + for the same purpose.
!	Logical negation (Not)

Supported Functions in Expressions in Cloud/Edge Processes

For a cloud/edge process, you can use some functions in the expressions to define parameters or conditions. Refer to the following table for the supported functions.

Function	Description
`var:lte(varName, value)`	Lower than or equals
`var:lt(varName, value)`	Lower than
`var:gte(varName, value)`	Greater than or equals
`var:gt(varName, value)`	Greater than
`var:get(varName)`	Retrieve the value of a local variable, using this function won't throw an exception when the local variable doesn't exist.
`var:getOrDefault(varName, defaultValue)`	Providing a default value which is returned when the local variable isn't set or the value is null.
`var:notEmpty(varName)`	Checks if the local variable value is `notEmpty`.
`var:empty(varName)`	The reverse operation of `isEmpty`
`var:eq(varName, value)`	Check if a local variable equals to a given value.
`var:ne(varName, value)`	The reverse operation of `equals`
`var:contains(varName, value1, value2, …)`	Check if all values provided are contained within a local variable.
`var:containsAny(varName, value1, value2, …)`	Similar to the `contains` function, but `True` will be returned if any (and not all) passed values are contained in the local variable.
`var:base64(varName)`	Convert a binary or string local variable in `Base64` string.
`var:exists(varName)`	Return `True` if the local variable has a non-null value.

Supported Functions in Expressions in Automation Sequences

For an automation sequence or a subscription, you can use some functions in the expressions to define parameters or conditions.

Refer to this link for the supported functions.

Manage Users and Work Groups in a Production Process Design

A work group is a group of users that work collectively as a unit. With work groups, you can manage users in production process designs or deployment groups more easily. You can delete a work group as long as it's not being used in any production process designs. Invite a user to work on a production process design by assigning the user or a work group that contains the user to the production process design.

Prerequisites

- You have the role of `Manufacturing_Admin`, `Manufacturing_Admin_Limited`, `Automation_Engineer` or `Production_Engineer`.
- You're assigned to this production process design.

  By default, the creator of a production process design is automatically assigned to it.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. On the **Users** tab, you can:
   - Add or remove users from the production process design.
   - Add work groups from the production process design. Select an existing work group or create a new one. Choose *(delete)* to delete work group in the **Select Work Groups** window. Choose *(expand)* to work group detail page. You can manage users in the work group and change work group name and description.

     > **Note**
     > As the creator of a work group, you are automatically assigned to it.

   - Remove work groups from the production process design.

Related Information

Manage Users in a Deployment Group
Superuser for Managing Users and Work Groups

Superuser for Managing Users and Work Groups

A user with the roles of `Manufacturing_Admin`, `Manufacturing_Admin_Limited`, `Production_Engineer`, or `Automation_Engineer` can manage users for production process designs, deployment groups, and work groups even though the user himself or herself is not assigned to them. This avoids the situation where the only authorized user is blocked from accessing the system for whatever reason (for example, the user has left the company) and the relevant objects in the system become unusable or user management for these objects become impossible.

The roles of `Manufacturing_Admin` or `Manufacturing_Admin_Limited` are not allowed to edit the relevant objects, such as updating a production process or adding shop floor elements to a deployment group. They are only for managing users in case of above-mentioned situations.

The special permissions for the roles of `Manufacturing_Admin`, `Manufacturing_Admin_Limited`, `Production_Engineer`, and `Automation_Engineer` are as follows:

App	User Management Actions
Design Production Processes	• Add users and work groups to a production process design • Remove users and work groups from a production process design • Add users to a work group • Remove users from a work group

App	User Management Actions
	• Delete work groups that are not assigned to any production process designs • Add more work groups, if necessary
Deploy Shop Floor Elements	• Add users to a deployment group • Remove users from a deployment group

Activity Logs in a Production Process Design

You can search and view the activity logs of the design time.

The system logs the activity you have performed in the design time of the production process design on the **History** tab. The log covers:

- Object name
- Object type
- ID
- Activity
- Date and time of the activity

  > **Note**
  > The system uses the time zone of your selected plant as the standard time. If no plant is selected, the time of your current browser will be displayed instead.

- The user performing the activity

Create a Production Process Design

A production process design is a group of production processes (for example, for one particular production line). Before creating production processes, create a production process design first.

Prerequisites

You have the role of `Automation_Engineer` or `Production_Engineer`.

Context

> **Note**
> The name and the version are both case-sensitive.

Procedure

1. Open the **Design Production Processes** app.
2. Choose **Create**.
3. In the **Create Production Process Design** window, specify the required information.
    a. **Name**: For your own memory purposes, enter a name. With characters of any language and a maximal length of 100, the name can be changed at any time for the production process design in draft or modified status. Except for archived production process designs, there must not be any production process designs with duplicated names in a single landscape.

    > **Note**
    >
    > The names of production process designs aren't case-sensitive. Names that use the same letters, regardless of upper or lower case, are treated as duplicates.

    b. **ID**: An ID is a unique identifier with limited western characters (A–Z, a–z, 0–9, _) for technical use. ID is automatically generated as you type the name. You can also define the ID by yourself.

    > **Note**
    >
    > ID can only be changed in the draft production process design.

    c. **Description**

    d. **Version**

    e. **Label**: Add a label with limited western characters (A–Z, a–z, 0–9, _) and a maximal length of 128 to categorize the production process design in this version. All production processes under this design inherit this label as well. Choose from an existing label or create a new one. You can also edit the label later for production process design in **Draft** or **Modified** status. Label information is retained when you copy, export, or import the production process design. You can easily filter out the production process design on the list page with the label you have added. You can also filter out process instances with this label in the **Monitor Production Processes** app. See Search and Filter Process Instances for details.
4. Choose the **Create** button.

> **Note**
>
> Choose **Edit Header** next to the production process design name on the detailed page to edit related information.
>
> Choose **Validate** to check any dependency issues that can block the deployment.

Next Steps

Manage Users and Work Groups in a Production Process Design

Update a Production Process Design

For each production process design, only one draft version is allowed. After a production process design is deployed or deleted, you can create a new version of it. Note that you do not have to be assigned to a production process design to save it as a new version.

Prerequisites

- You have the role of `Automation_Engineer` or `Production_Engineer`.
- The production process design is in any of the following statuses:
    - **Deployed**
    - **Archived**

Context

All production processes are reserved in the new version. You can modify or delete the production processes.

All the users assigned to the previous version are still assigned to the new version. If you were not assigned to the previous version, you are now assigned to the new version.

You can select different versions of production process designs in the **Version** dropdown list inside the production process design detailed page.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. Choose **Edit**.
4. Add your own name or leave it as is.
5. Use the suggested version or enter a new version.
6. Choose **Save as New Version**.

Results

The status of the new version is **Modified**.

When updating a production process design (in **Archived** or **Deployed** status) with changes, as its production process may have already been used in other places, such as another production process or automatic trigger, you will be reminded to check the where-used list before deployment. You can either give up your changes or continue with them.

ℹ Note

If you change the input or output parameters of a production process used by a subscription and then deploy its production process design, the subscription will get updated with this change. If the subscription is already deployed, a modified version will be auto-generated by the system. You need to check the subscription and reassign the parameter value accordingly.

When you deploy a new version of a production process design, the runtime configurations of the previous version will be updated automatically to that of the new version in the runtime environment. Meanwhile, the status of the previous version becomes **Archived**.

ℹ Note

You cannot submit modified production process design to deployment group if its previous version is in **Awaiting Un-deployment** status. You need to remove the previous version from the deployment group created for un-deployment to proceed.

Copy a Production Process Design

You can create a new production process design by copying an existing one. Note that you do not have to be assigned to the production process design to copy it.

Prerequisites

You have the role of `Production_Engineer` or `Automation_Engineer`.

Context

All production processes are reserved in the new production process design. You can modify or delete the production processes or create new ones.

> **Note**
> The name and ID for the new production process design must be different from the old one.

All the users assigned to the copied production process design are still assigned to the new one. If you were not assigned to the copied production process design, you are now assigned to the new one.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. In the additional options dropdown menu, choose **Copy** .
4. Enter a new name and ID that is also unique.
5. Optionally, change the description, type or version.
6. Choose **Copy**.

Delete Production Process Designs

When a production process design is in the **Draft**, **Modified**, **Archived** status, you can always delete it. After deploying the production process design from the cloud to the runtime environment, you must first delete the design from the runtime environment (un-deploy) and then from the cloud.

Prerequisites

- You have the role of `Automation_Engineer` or `Production_Engineer`.
- You're assigned to this production process design.

Delete a Deployed Production Process Design from the Runtime Environment

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design in the status of **Deployed**.
3. Choose **Select Deployment Group** in the dropdown menu of **Quick Delete**.

> **Note**
> For a fast deletion, you can choose **Quick Delete** to complete the following steps all at once (while still need to delete it from cloud afterwards).

4. In the **Add to Deployment Group** window, confirm the default deployment group automatically created and choose **Submit for Deployment**. The status of the production process design becomes **Awaiting Un-deployment**.

   Alternatively, instead of using the default deployment group, you can select other existing deployment groups by choosing **Select Another Group**. In the **Select Deployment Group** window, you can also choose *(add)* to create a new deployment group. If you create a new deployment group, you become its administrator automatically.

5. In the Deployment Group window, adjust the shop floor elements that you want to un-deploy.
6. Choose **Deploy and Activate**.

> **Note**
>
> You can still go to the **Deploy Shop Floor Elements** app to un-deploy the production process design afterwards. See Deploy and Activate Shop Floor Elements for details.

Results

- All runtime configurations are deleted.
- The status of the production process design becomes **Archived**.

Delete from the Cloud

Procedure

1. Open the **Design Production Processes** app.
2. Choose *(Filter)* and select **Archived**.
3. For a production process design in the status of **Archived**, choose *(Delete)*.
4. Confirm the deletion.

Delete in Batch

Procedure

1. Open the **Design Production Processes** app.
2. Select the production process designs that you want to delete in batch.
3. Choose **Delete**.

> **Note**
>
> Designs in the **Draft**, **Archived**, **Modified** status will be deleted directly. Designs in the **Awaiting Un-deployment**, **Awaiting Deployment** or **Failed** status will be ignored. Deployed designs will be submitted to a deployment group for your further action.

Export and Import Production Process Designs

You can export production process designs with cloud processes and relevant services (user-defined services and MII services) to a file, and upload to other systems.

Prerequisites

You need to have the role of `Production Engineer` to export and import.

⚠ Caution

Transferring to a production system may introduce uncertain risks because the data in the production system changes. You are responsible for any changes you make. Therefore, we recommend transferring to a non-production system instead. If you're still transferring from a quality system to a production system, it's highly recommended to ensure both systems are on the same release. Avoid performing the transport during quality testing phases because system versions differ during that period. Backward transport is only used in special circumstances. Please avoid frequently performing transport between multiple systems. The best practice is to always follow the transport in one direction, such as from the development system to the quality system and from the quality system to the production system.

It's highly recommended that you use the same user throughout the entire transport process in the Production Process Designer of both the source and target systems for all operations, including any further modifications. This helps prevent import failures due to authentication issues.

For version management, refer to the relevant section in User and Version Management.

Export

Procedure

1. Open the **Design Production Processes** app.
2. Select the production process designs you want to export.
3. Choose **Export**.
4. Set a password.

   Only production process designs with cloud processes and relevant services can be exported. You can choose **Continue Anyway** or **Close** for production process designs that do not meet the criteria.
5. Choose **OK**.

   You can see the export status in the message strip on the list page. Once the export is finished, you can choose **Download** to download the file. The status of production process designs will be changed to draft after the export.

   You can choose **Check Details** for detailed information about the export.

   - Choose **Skip and Download** to skip exporting failed production process designs and continue exporting other ones.
   - Choose **Retry All** to redo the export for all designs you have selected.
   - Choose **Close** to stop the export.

ℹ Note

The detailed information is kept for seven days and will not be available afterwards.

Import

Procedure

1. Open the **Design Production Processes** app.
2. Choose **Import**.
3. Choose the local file that you have downloaded from the source system.
4. Enter the password you have set.
5. Choose **Continue**.

   You can see the import status in the message strip on the list page and choose **Check Details** to open the import manager for detailed information about the import.

   Production process designs without import errors are displayed in the **In Progress** tab, with their import status information included. The URL/path of the published production processes in the production process design stays the same as the one in the source system.

   Production process designs with import errors are displayed in the **Wait for Action** tab of the import manager in the following three categories:

   - **Duplicated Design**: There is already the same design (same ID, same or different name, imported before) or a design with the same name (same name, same or different ID, not imported before) in the target system. Choose **Import a Copy** to import it as a separate one and add a new name. Meanwhile, a new ID will be auto-generated. You can also **Overwrite** the existing design with the new one (ID won't change, still the same ID as the design of the target system). **Skip** will ignore the selected design.

     > **Note**
     >
     > If the production process in the production process design has been published to the **Manage Service Registry** app, its URL will change when you choose Import a Copy. For the overwrite option, the URL won't change.
     >
     > For the overwrite option, if the design of the target system is in deployed status, it stays as is. A new modified version (with the same ID as the design of the target system) gets added to it. If the design of the target system is in draft or modified status, it gets overwritten by the imported design (with the same ID as the design of the target system).

   - **Other Error**: All designs with other errors except the above ones are displayed here. Choose **Show Detail** for further error descriptions. You can choose **Skip** to ignore the selected design.

   You can view all production process designs imported by you in the **My Imports** tab on the list page.

Next Steps

After you import the production process design, you see services in the production process get automatically added the **Manage Service Registry** app. However, the web server information is missing. To get back the information, create a web server that has the same name as the one in your source system in the **Manage Web Servers** app.

Transport Production Process Designs Through SAP Cloud Transport Management

Transport SAP Digital Manufacturing configuration objects across different systems by integrating SAP Digital Manufacturing with SAP Cloud Transport Management service.

Prerequisites

- You are a global account administrator.

- You have set up your global account and at least one subaccount on SAP Business Technology Platform. For an overview of the required steps, see Getting Started in the Cloud Foundry Environment.
- You have performed the initial setup steps to be able to use SAP Cloud Transport Management service. See the initial setup steps in Set Up the Environment to Transport Content Archives directly in an Application for details.

> **Note**
>
> To set up the SAP Cloud Transport Management service, you only need to do the initial setup and subscribe to the SAP Cloud Transport Management service on any of your subaccounts, even if this subaccount is not your source or target tenant account for transporting data.

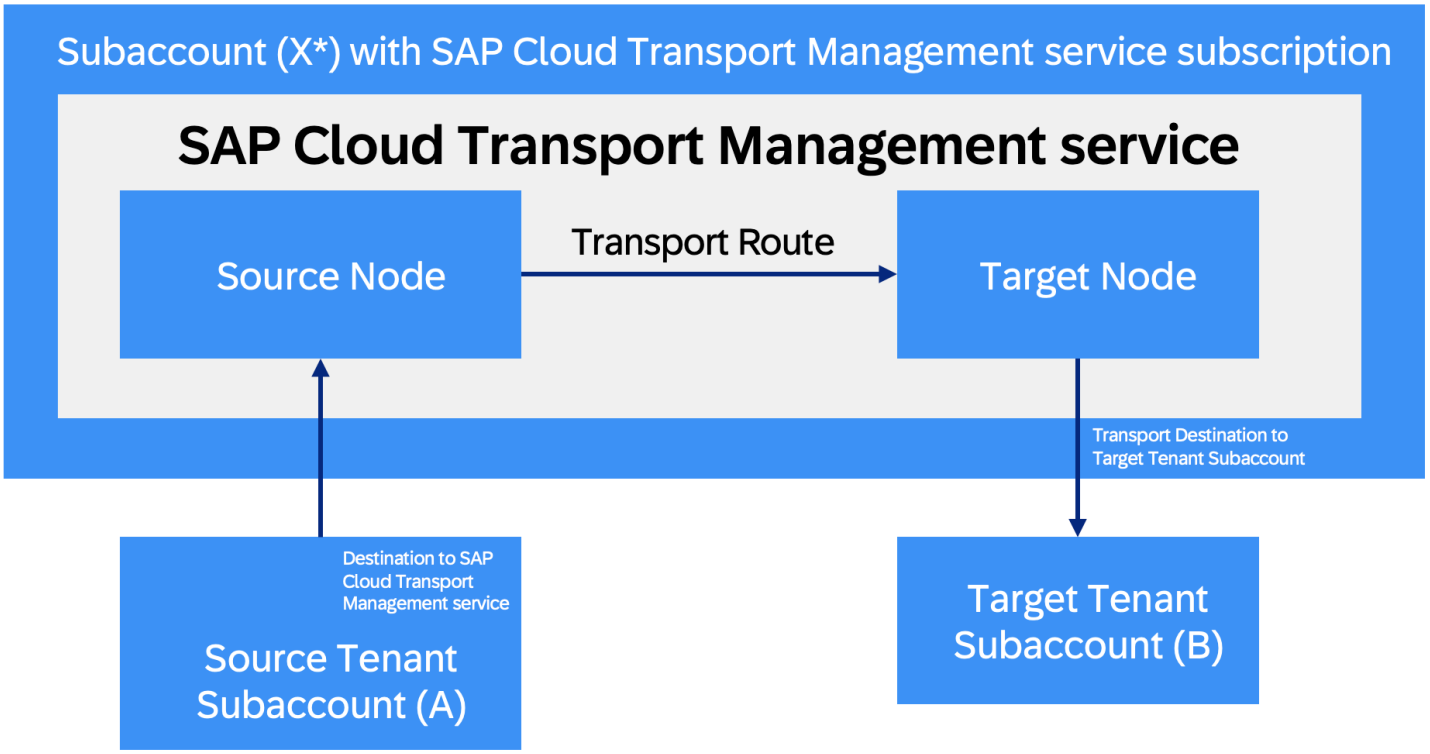
- You have the role of `Production_Engineer`.

Context

This integration allows you to seamlessly transfer production process designs between systems. Transport nodes represent source or target endpoints of your subaccount system.

> **Caution**
>
> Transferring to a production system may introduce uncertain risks because the data in the production system changes. You are responsible for any changes you make. Therefore, we recommend transferring to a non-production system instead. If you're still transferring from a quality system to a production system, it's highly recommended to ensure both systems are on the same release. Avoid performing the transport during quality testing phases because system versions differ during that period. Backward transport is only used in special circumstances. Please avoid frequently performing transport between multiple systems. The best practice is to always follow the transport in one direction, such as from the development system to the quality system and from the quality system to the production system.
>
> For user management, refer to the relevant section in User and Version Management.

> **Note**
>
> Transporting production process designs with cloud processes and relevant user-defined services is supported. However, SAP Digital Manufacturing for edge computing production processes and automation sequences of Production Connector on Windows/edge runtime environment are not supported for transportation.



**X can be either A, B, or any other subaccount.*

Procedure

1. Prepare task in the subaccount where you have subscribed to the SAP Cloud Transport Management service.

    a. Create SAP Cloud Transport Management service instance and service key. You need to note down the `clientid`, `clientsecret`, `url` and `uri` of this service key to configure later the destination to SAP Cloud Transport Management service in the source tenant subaccount from which the production process designs to be exported. See Create a Service Instance and a Service Key for details.

    b. Create transport destinations to the target tenant subaccount with the service key of SAP Digital Manufacturing services of the target system. Transport destinations are used to address the target endpoint of an import process. See Create Transport Destinations for details.

    > ### ℹ Note
    > You need to have already noted down the `clientid`, `clientsecret`, `url` and `public-api-endpoint` of the service key of SAP Digital Manufacturing services of the target system (different from the service key in step 1.a). Refer to Prepare for API Integration on how to get the service key of SAP Digital Manufacturing services.

Property	Value
Name	Define a name on your own.
Type	HTTP
Description	Not required.
URL	Enter the `public-api-endpoint` in the service key and add `/transport` to the end. Refer to Prepare for API Integration for details on the service key of SAP Digital Manufacturing services.
Proxy Type	Internet
Authentication	OAuth2ClientCredentials
Client ID	Service key: `clientid`

Property	Value
	Refer to Prepare for API Integration for details on the service key of SAP Digital Manufacturing services.
Client Secret	Service key: `clientsecret` Refer to Prepare for API Integration for details on the service key of SAP Digital Manufacturing services.
Token Service URL	Enter the `url` in the service key and add `/oauth/token` to the end. A typical token service URL, for example, should look like `https://{$Subdomain}.authentication.xxxx.hana.ondemand.com/oauth/token` Refer to Prepare for API Integration for details on the service key of SAP Digital Manufacturing services.
Token Service User	Not required.
Token Service Password	Not required.

2. Prepare task in the source tenant subaccount. Create a destination to SAP Cloud Transport Management service with the following information:

Property	Value
Name	SAP_DM_TRANSPORT_SERVICE
Type	HTTP
Description	Not required.
URL	Service key: `uri` Refer to the information collected in the previous step.
Proxy Type	Internet
Authentication	OAuth2ClientCredentials
Client ID	Service key: `clientid` Refer to the information collected in the previous step.
Client Secret	Service key: `clientsecret` Refer to the information collected in the previous step.
Token Service URL	Enter the `url` in the service key and add `/oauth/token` to the end. A typical token service URL, for example, should look like `https://{$Subdomain}.authentication.xxxx.hana.ondemand.com/oauth/token` Refer to the information collected in the previous step.
Token Service User	Not required.
Token Service Password	Not required.

3. Configure in SAP Cloud Transport Management.

a. Create a source node that represents your source tenant subaccount. This is where the production process designs are exported. Use the destination to SAP Cloud Transport Management service that you created in step 2. Create a target node that represents your target tenant subaccount. This is where the production process designs are imported. Use the transport destination that you created in step 1.b. Choose **Application Content** as the content type when creating the target node. See Create Transport Nodes.

	Source Node	Target Node
Upload Allowed	Yes	No
Perform Notification	No	No
Content Type	Application Content	Application Content
Destination	SAP_DM_TRANSPORT_SERVICE	Destination created in step 1.b
Forward Mode	Pre-Import	Pre-Import
Virtual Node	No	No

b. Create transport routes from your source transport node to target transport nodes. See Create Transport Routes.

4. Send transport requests from the Production Process Designer to SAP Cloud Transport Management.

a. On the list page of the **Design Production Processes** app, select the production process designs that you want to transfer to the target nodes you have set in SAP Cloud Transport Management.

> **Note**
>
> You can select up to 20 production process designs in each transport request. Ensure that the current user of your production process designs is also in the work group of production process designs in the target system.
>
> For version management, refer to the relevant section in User and Version Management.

b. Choose **Transport** in the list header.

c. In the **Transport** window, provide the following information:

- **Source Node**: Source node you have set in SAP Cloud Transport Management. Only source nodes mapped to a target node in the transport route are visible here.
- **Description**: Add some text description for this transport. With a length no longer than 512 characters, the description can only contain A–Z, a–z, 0–9, spaces, and the following characters: `' - . _ ~ : \ / ? # [ ] @ ! $ & + , ; =*`

d. Choose **Confirm**. You can view transport request details in the message strip at the top of the list page after creating the transport request.

5. Import to the target nodes. For details, see Using the Import Queue.

a. Log into **SAP Cloud Transport Management** tenant using your credentials with the **Check details** link in the message strip of the **Design Production Processes** app to verify your export.

b. In the **Transport Request** detail page, choose the target node to which you'd like to import the production process designs. You can view transport request content details in the **Content** tab of the **Transport Request** detail page.

c. Under the **Import Queue** tab of the **Transport Nodes** detail page of target node, you can see the transport request of your production process designs ready for import.

d. Select the transport request and choose **Import Selected**. The selected production process designs in the transport request are transported to the target system. You can also enable immediate imports of transport requests as soon as they enter a new queue, or run imports of all transport requests in a transport node regularly. For details, see Enable Automatic Import and Schedule Imports.

Results

Once the import is completed, you can see the content of the production process designs available in the **Design Production Processes** app of its target system.

You can check the transport action logs in case the import fails.

Related Information

User and Version Management

User and Version Management

This topic explains how users and versions are managed when you transfer production process designs between systems. The version management section applies to transfers using the export and import feature of the Production Process Designer and through SAP Cloud Transport Management. However, the user management section only applies to transfers through SAP Cloud Transport Management.

User Management

It's highly recommended that you use the same user throughout the entire transport process in SAP Cloud Transport Management, Production Process Designer of both the source and target systems for all operations, including any further modifications. This helps prevent import failures due to authentication issues. However, if you're still using different users, the target system recognizes only the user who conducted the import in SAP Cloud Transport Management tenant as the user of the imported production process design. When you use the automatic or scheduled import feature in SAP Cloud Transport Management, the target system regards a technical user, auto-generated in this process, as the user of the imported production process design.

Source System	SAP Cloud Transport Management	Target System (after import)
User A	User A	User A
User A	User B	User B
User A	• User B (login account) • Technical User (CTMS_APPLICATION for automatic import, IMPORT_SCHEDULER for scheduled import)	Technical User

Version Management

When you do the import for the first time, the production process design version in the target system matches the version from the source system. For each subsequent import, if the production process design in the target system is in deployed status, along with this existing one, the version of the newly-imported one in the target system increases by 0.1, no matter the version from the source system. However, if the production process design in the target system is in draft or modified status, the version in the target system follows the version in the source system with each import. This rule also applies to the export and import features of the Production Process Designer.

Source System	Target System (before import)	Target System (after import)	Note
PPD1 (1.1, draft/deployed)	/	PPD1 (1.1, draft)	initial import
PPD1 (1.3, modified/deployed)	PPD1 (1.1, draft/modified)	PPD1 (1.3, modified)	non-initial import
PPD1 (1.3, modified/deployed)	PPD1 (1.1, deployed)	PPD1 (1.2, modified) PPD1 (1.1, deployed)	non-initial import

If there is already the same design (same ID, same or different name, imported before) in the target system, the system will overwrite the existing design with the new one (ID won't change, still the same ID as the design of the target system). If there is already a design with the same name (same name, same or different ID, not imported before) in the target system, an error will occur. You need to change the name of the design to be imported and try again. Note that this case is only for initial imports.

The URL or path of published production processes in the production process design remains unchanged from the source system. If published production processes already exist in the target system and are deployed, the URL or path remains unchanged from the target system.

Source System	Target System (before import)	Target System (after import)	User Action	Note
PPD1 (ID1) - PP1 (draft/deployed, URL1)	/	PPD1 (ID1) - PP1 (draft, URL1)		initial import
PPD1 (ID2) - PP2 (draft/deployed, URL2)	PPD1 (ID1 or ID2) - PP1 (draft/deployed, URL1) **Note** In this case, you have created this production process design directly in the target system instead of importing it.	error	Change name "PPD1" to "PPD2".	initial import
PPD1 (ID1) - PP1 (draft/deployed, URL1)	PPD1 (ID1) - PP1 (draft, URL1)	PPD1 (ID1) - PP1 (draft, URL1, content from source system PP1)		non-initial import
PPD1 (ID1) - PP1 (draft/deployed, URL1)	PPD1 (ID1) - PP1 (deployed, URL1)	1. PPD1 (ID1) - PP1 (deployed, URL1) 2. PPD1 (ID1) - PP1 (modified, URL1, content from source system PP1)		non-initial import

*PPD = production process design; PP = production process

Related Information

Export and Import Production Process Designs
Transport Production Process Designs Through SAP Cloud Transport Management

Create and Define a Production Process

Create a production process by defining a sequence of activities based on rules. The activities are driven by various kinds of services that are either available in the cloud (including those provided by SAP Digital Manufacturing itself), edge or provided by an on-premise system.

Prerequisites

- To create, run, update and delete a cloud/edge process, you must have the `Production_Engineer` role.
- To create, run, update and delete an automation sequence, you must have the `Automation_Engineer` role.
- To define error end events and corresponding error catch events for cloud/edge processes, define suitable error codes.

  For more information, see Define Error Codes for Error End Events.
- See Available Services and Subprocesses for other prerequistes in detail.

Context

Depending on the runtime environment, the production processes are divided into three categories: cloud processes that are run in the cloud, edge processes that are run in the edge, and automation sequences that are run by a Production Connector system. When creating a production process, you must first specify a runtime environment.

The different categories of production processes have different controls and services available for use.

A production process can embed other production processes for more flexible reuse. There is one limitation for the automation sequences: the parent and child (embedded) processes must be run by the same Production Connector system. Also, edge process can embed only production processes from the same edge runtime instance.

Related Information

Controls
Available Services and Subprocesses
Local Variables
Supported Operators in Expressions
Data Types and Data Type Conversion

Create a Production Process

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. Choose *(Create)*.
4. In the **Create Production Process** window, enter the required information.
   - **Name**: For your own memory purposes, enter a name. While names can be duplicated with characters of any language and a maximal length of 100 across production process designs, there must not be any production processes with duplicated names in a single production process design.

- **ID**: An ID is a unique identifier with limited western characters (A–Z, a–z, 0–9, _) for technical use. ID is automatically generated as you type the name. You can also define the ID by yourself.

  > **Note**
  > ID can only be changed in the draft production process, but the name can be changed at any time for the production process in draft or modified status.

- **Runtime Type**: Select either **Cloud**, **Edge**, **Production Connector**, **Plant Connectivity**.
- **Runtime Environment**: Choose the runtime web server in which the production process will be deployed.

  > **Note**
  > If you choose Production Connector as the runtime type, select the web server of Production Connector on Windows or Production Connector on edge as the runtime environment to create an automation sequence.

- **Visible to Production Connector / Plant Connectivity Runtime, Visible to Cloud/Edge Runtime**: Check to make the production process you created visible to other runtime. Alternatively, you can configure it later by editing header. See Available Services and Subprocesses for details.

5. Choose **Create**.

> **Note**
> Choose **Edit Header** next to the production process name on the detailed page to edit related information.

Define a Production Process

Procedure

You may not follow the steps below in a strictly sequential order. They are documented in such a way that you can have an overview of all the steps required for defining a production process.

1. Open a production process in the editor view.

   If the process is not already open, click the process in the repository view () and then go to the editor view ().

2. Drag and drop required elements - controls, services and processes - to the canvas, within a swimlane.

   Use Shift to select more than one element of the process.

   When you embed a process into another process, the former process becomes a subprocess of the latter process.

   > **Note**
   > By default, the system places the start and end elements in the canvas.
   >
   > Some services are by default hidden. Edit the service library to display them, if required. For available services to different runtimes, see Available Services and Subprocesses.
   >
   > Services are organized in swimlanes by web server names.

For edge runtime, you can only see the services available on selected edge instances.

3. Select a destination from web server or input a secret name for third-party services. For details, see Use Third-Party Service in Cloud/Edge Process.
4. Connect the elements with lines.
5. If required, define evaluation expressions for the condition elements.

> **Note**
>
> If it is a cloud/edge process, you can drag the expressions up and down to change the order of evaluation.

6. If required, create local variables for parameter mapping.

> **Note**
>
> Parameter data type of **Start** element should match the local variable data type.

7. If required, create process input parameters on the **Start** control and define the parameter values.

> **Note**
>
> The parameter or local variable name can only contain A–Z, a–z, 0–9 and underscores. It must start with a letter and end with either a letter or a number. Double underscore is not allowed. Some reserved words are not allowed. For details, see 3379404.
>
> Choose **Manage Parameters** for batch editing of parameters. You can check the **Required** box to set the value of parameter as mandatory for cloud/edge processes.
>
> You can use the "Execute Asynchronously" option of cloud/edge processes or automation sequences (run as subprocesses) to prevent the execution result from affecting the main process. This asynchronous execution of subprocesses is displayed separately in the **Monitor Production Processes** app.
>
> See Data Types and Data Type Conversion for supported data types in local variable, global variable, and parameter values.
>
> To designate complex data type (schema defined in service registry) to local variable, global variable or input/output parameter value of cloud/edge production process and automation sequence, script task and service, choose `Structure` or `StructureArray` as type during creation. For details on adding parameter or variable values, see Add Value for Parameter or Local Variable.

8. If required, create process output parameters on the end element and define the parameter values.
9. Define the input and output parameter values of all services and subprocesses.

> **Note**
>
> For third-party services coming from service registry, only `String` enum of parameter or `String` item enum of array parameter can be consumed in Production Process Designer.
>
> When the service responses multiple codes, only response code 200 response schema will be shown in the design side.
>
> Making changes to a service or schema may affect its usage in the production process. A compatible change won't cause runtime errors while an incompatible one will do. For example, changing the name and description, adding optional parameters and response code, are compatible service changes, while changing the API path, adding mandatory parameters, and changing HTTP method, web server, or response code, are incompatible service changes.
>
> The service name length should not exceed 150 characters.
>
> You will see an □ *(error)* next to the service or subprocess if it has been deleted or disabled. You can remove it or replace it with another one. In other cases, you will see a □ *(warning)* if the service or subprocess has a compatible change.

Choose **Synchronize** in the message strip or the □ *(synchronize)* in canvas to refresh it. If the change is incompatible, you will see an □ *(error)*. Choose □ *(synchronize)* to refresh it and reassign the values. If the change is about a general information change, such as URL, path, HTTP method, or protocol, you will also see an □ *(error)*. These changes are refreshed automatically when you modify the production process.

For schema (complex data type) used in services, script tasks, the input or output parameters of the production processes, or local variables and global variables in production processes, its compatible changes, for example, can cover changing property description, or enum values, while adding mandatory properties, changing the property name and data type are incompatible changes. If the schema has a compatible change under the **Schemas** tab of the **Manage Service Registry** app, you will see a □ *(warning)*. If the change is incompatible, you will see an □ *(error)*. Redo the value mapping to make sure that the problem is solved. Choose **Done** and **Complete All to Proceed**.

You can review all the above errors or warnings in an issue report in the production process header.

In a production process of cloud or edge runtime, if a user-defined service has multiple content types defined for request body and response, you can choose **application/json** or **text/plain** in **INPUT/OUTPUT** to define the input/output parameter in JSON, or plain text. You also have the option to add local variables or parameters to the input parameter area of the text/plain type service with the **Insert** button. Note that if text/plain is defined as content type in response in the **Manage Service Registry** app, only string data type is allowed in the Production Process Designer.

In the **Automatic Retry Settings**, define the time and interval for the service (RESTful and OData) to be retried if there is a failure during execution. From the dropdown list, select the error codes by which you expect the retry to be triggered. Add an interval range from 1 to 60 seconds and a retry limit of no more than 30 times. If the retry interval is less than 5 seconds, the retry limit is allowed at most 3 times. Note that a system delay within 10 seconds might add to the actual interval time in the runtime environment. Execution performance will be affected if the retry limit is over 10 times. The HTTP status codes cover standard response codes that you can refer to Standard Response Codes of Service and Production Process.

Besides the automatic retry settings you have configured, the system performs automatic retry by default in the following situations:

API Method	Service (RESTful and OData)	Registered Production Process / Read Indicators, Read Indicators Flexibly, Read Indicators Locally
GET	a. error codes: 500 or above b. all connection or timeout issues without error codes c. error messages: i. recvAddress (..) failed: Connection reset by peer ii. Failed to resolve	/
POST, PUT, PATCH, DELETE, or methods other than GET request	a. error messages: i. readAddress (..) failed: Connection reset by peer ii. Connection refused iii. Connection timed out	a. error codes: above 500 b. all connection or timeout issues without error codes c. error messages: i. recvAddress (..) failed: Connection reset by peer

API Method	Service (RESTful and OData)	Registered Production Process / Read Indicators, Read Indicators Flexibly, Read Indicators Locally
	iv. writeAddress (..) failed: Connection reset by peer v. recvAddress (..) failed: Connection reset by peer vi. Failed to resolve b. error code: 503 (service unavailable)	ii. Failed to resolve

10. You can directly pass input parameters of the main production process or output parameters of previous service/subprocess to the value of input parameters of late service/subprocess, or output parameters of the main production process. The output parameter value of services in cloud/edge processes can also be written back to local variables.

    ### ℹ Note

    Output parameters of subprocesses with "Execute Asynchronously" checked will not be passed to the value of parameters of late elements.

    To pass complex data type values, you can configure the mapping relations when adding the input parameter values of the late service.

11. You can publish deployed cloud/edge processes to service registry by choosing **Edit Header** and switching on the **Publish to Service Registry** toggle. The process you publish will be displayed in service library for easy reuse in other apps.

    ### ℹ Note

    You can configure **Publish to Service Registry** during the design process, but the option takes effect only after the production process design is deployed.

    Production process published to the service registry, third-party service, or SAP Digital Manufacturing cloud service is executed synchronously by default if it is called by the main process in the Production Process Designer. Alternatively, you can define it in the “async” query parameter for all production processes published to the service registry and some supported third-party services, or SAP Digital Manufacturing cloud services.

12. The Production Process Designer provides you the option to use a production process as a subprocess in the other runtime in the same production process design. For automation engineer, you can grant production engineer permission to call automation sequence on Production Connector. For production engineer, you can grant automation engineer permission to call production process on cloud. To do so, make sure SAP Digital Manufacturing services type web server has already connected to the Production Connector web server and open the production process first. Then, choose **Edit Header** and switch on the **Visible to Cloud/Edge Runtime** or **Visible to Production Connector / Plant Connectivity Runtime** toggle. Save it and now you can see them displayed under the repository list of the left panel in the corresponding runtime.
13. You can also expose the automation sequence as a RESTful service to be called by assets or systems on the shop floor. For details, see Expose Automation Sequence as RESTful Service.
14. When you edit or delete a production process that has been used in other places, the system will detect it automatically and you can check in the warning message box where this production process is used. You can continue with your update or cancel. Alternatively, you can also choose **Check Where-Used** in the production process detail page to check where the production process has been used. Refresh the list to see the latest changes if the production process gets updated.

    Besides the general information about the consumers provided in the where-used list, the usage status indicates if the production process is used in an up-to-date manner by its consumers. You should pay attention and take certain actions to those production processes with “outdated” usage status. Possible usage status:

- **Normal**: The production process is being used normally by the consumer.
- **Outdated (compatible)**: The production process is outdated in the consumer due to compatible changes. Recommend checking further.
- **Outdated (incompatible)**: The production process is outdated in the consumer due to incompatible changes. You need to synchronize the production process to keep using it.

Navigate directly to each consumer through its link and check the production process accordingly.

Add Value for Parameter or Local Variable

Parameter or local variable value in the **Design Production Processes** and **Manage Automatic Triggers** app supports primitive, primitive array and complex data types.

There are six ways to add values:

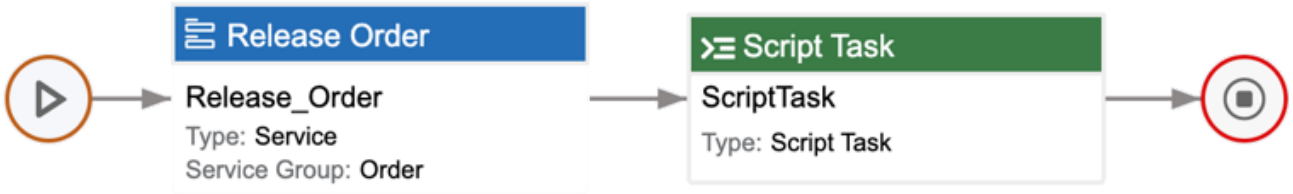
1. **Direct input**: Input directly constant value, local variable, or another parameter with the same data type, structure, or expression.
2. **Expression editor**: Choose **Open expression editor**. You can use the expression editor to input constant values and expressions. The expression editor is only supported when the service input parameter is of primitive data type. Note that in the cloud/edge process, the input/output parameters of script task and output parameters of end element do not support expression.
3. **Dropdown**: Choose one of the local variables or other parameters that are of the same primitive or primitive array data type as the current parameter. In other cases, choose one of the complex data type variables with the same structure as the current parameter.
4. **Select structure property**: Choose the value help icon and in the **Select Structure Property** window, choose from complex data type one property whose data type is compatible with the current parameter. In other cases, choose from primitive array data type one item whose data type is compatible with the current parameter. Then choose **Save**.
5. **Input value**: Choose the value help icon and in the **Input Value** window, choose one of the complex / primitive array data type local variables or other parameters with the same structure / item as the current parameter. The system will map all sub-properties / items automatically. If your selection is partially matched to the current parameter, the matched sub-properties / items will be mapped automatically while a warning icon will be displayed in the **Compatibility Check** column for incompatible ones. Meanwhile, the parent property / primitive array data type you have originally chosen will be cleaned. The system supports two editing modes: form view and code view. Switch between modes by choosing **Switch to Code** or **Switch to Form**.
6. **Assign output** (for subscription or business rule): Choose the value help icon and in the **Assign Output** window, decompose the complex / primitive array data type and assign properties to local variables.

ℹ Note

The Production Process Designer does not support writing the current parameter value back to a property in the complex data type local variables, an item in the primitive array type local variable, or partially matched complex / primitive array data type local variables. However, in case you still want to do so, you can use the script task to decompose the structure of the target local variable.

Example:

1. Add a script task element next to the current element whose output parameter (`Release_Order`) value you want to assign.



2. Create both input and output parameters (`startSfcRequest`) for the script task element with the very same data type as the target local variable (`var_sfcStartRequest`).
3. For the script task input parameter, choose the value help icon.
4. In the Input Value window, choose output parameter `Release_Order` in the Value From column and check the mapping.
5. For the script task output parameter, choose the local variable `var_sfcStartRequest`.
6. For the script, simply write a line to hand over the input parameter to the output parameter. For example, `$output.startSfcRequest = $input.startSfcRequest`.

		Primitive / Primitive Array Data Type		Complex Data Type	
Runtime	**Object**	**Get value from local variable or parameter**	**Write back to local variable**	**Get value from local variable or parameter**	**Write back to local variable**
Production Process (cloud, edge)	Web Service, Script Task, Production Process, Automation Sequence	For input parameters of service, script task, production process or automation sequence and output parameters of end element. primitive: [1], [2], [3], [4] primitive array: [1], [5]	For output parameters of service, script task or production process or automation sequence and input parameters of start element. primitive: [1], [3] primitive array: [1], [3] [Note]	For input parameters of service, script task, write indicator flexibly, production process, or automation sequence and output parameters of end element. [1], [5]	For output parameters of service, script task, read indicator flexibly, production process, or automation sequence and input parameters of start element. [1], [3] [Note]
Automation Sequence (Production Connector on Windows, Production Connector on edge)	Web Service, Production Process, Automation Sequence	For input parameters of service, production process, or automation sequence and output parameters of end element. primitive: [1], [2], [3], [4] primitive array: [1], [5]	For output parameters of service, production process, or automation sequence and input parameters of start element. primitive: [1], [3] primitive array: [1], [3]	For input parameters of service, production process, or automation sequence and output parameters of end element. [1], [5]	For output parameters of service, production process, or automation sequence and input parameters of start element. [1], [3]
	Shop Floor Service, Functions	For input parameters of	For output parameters of	/	/

		Primitive / Primitive Array Data Type		Complex Data Type	
Runtime	**Object**	**Get value from local variable or parameter**	**Write back to local variable**	**Get value from local variable or parameter**	**Write back to local variable**
		service and output parameters of end element. primitive: [1], [2], [3], [4] primitive array: [1], [5]	service and input parameters of start element. primitive: [1], [3] primitive array: [1], [3]		
Subscription (Production Connector on Windows, Production Connector on edge)	Web Service, Production Process, Automation Sequence	For input parameters of service, production process, or automation sequence. primitive: [1], [2], [3], [4] primitive array: [1], [5]	For output parameters of service, production process, or automation sequence. primitive: [1], [3] primitive array: [1], [6]	For input parameters of service, production process, or automation sequence. [1], [5]	For output parameters of service, production process, or automation sequence. [1], [6]
	Shop Floor Service	For input parameters of service. primitive: [1], [2], [3], [4] primitive array: [1], [5]	For output parameters of service. primitive: [1], [3] primitive array: [1], [6]	/	/
Timer	Web Service (including Registered Cloud Process)	For input parameters of service or production process. primitive: [1] primitive array: [1], [5]	/	For input parameters of service or production process. [1], [5]	/
Business Rule (Business Service)	Web Service (including Registered Cloud Process)	For input parameters of service or production process. primitive: [1], [2], [3], [4] primitive array: [1], [5]	For output parameters of service or production process. primitive: [1], [3] primitive array: [1], [6]	For input parameters of service or production process. [1], [5]	For output parameters of service or production process. [1], [6]
Business Rule (Event)	Registered Cloud/Edge Process	For input parameters of production process.	/	For input parameters of production process.	/

		Primitive / Primitive Array Data Type		Complex Data Type	
Runtime	**Object**	**Get value from local variable or parameter**	**Write back to local variable**	**Get value from local variable or parameter**	**Write back to local variable**
		primitive: [1], [2], [3], [4] primitive array: [1]		[1], [5]	

ℹ Note

In `String` data type parameters, enclose your input with double quotes for constant values or single quotes for local variables.

When assigning an array to string data type input parameters, if the array index is a local variable, you need to enclose it with single quotes. For example, `'input['a']'`. Note that for these cases the local variable data type for the array index supports `Integer`, `Short`, and `Long` only.

Because there is a situation that "." is inside of the property name, the Production Process Designer supports hybrid use of "." or "[]" as complex data type property access operator to distinguish same-level local variable reference from hierarchical reference. For example, `s.filter.material` refers to "material" under "filter". `s["filter.material"]` refers to "filter.material" under "s".

User-defined schema of array-of-array structure in service registry, `[[1,2,3],[4,5,6]]`, for example, is not supported in Production Process Designer.

The following special characters are supported in the string and string array data type (including string property of structure and structure array) input/output parameter value when you call RESTful services, including SAP Digital Manufacturing cloud service, read/write indicators, read/write indicators locally, read/write indicators flexibly, read/write global variables, script task, third-party services (RESTful), registered cloud processes, and automation sequences. Condition property and its value, action input/output parameter of business rule, the indicator and message payload property alias of the subscription in the **Manage Automatic Triggers** app, and parameter/variable value filter in the **Monitor Production Processes** app also support these special characters.

Special Character	Formal Name	Unicode Code Point	Note
#	Number Sign	U+0023	
$	Dollar Sign	U+0024	
/	Solidus	U+002F	
.	Full Stop	U+002E	
"	Quotation Mark	U+0022	Only supported in local variable value. Using in constant value is not supported. Not supported in the indicator and message payload property alias of the subscription.
'	Apostrophe	U+0027	Only supported in local variable value. Using in constant value is not supported. Not supported in the indicator and message payload property alias of the subscription.
@	Commercial At	U+0040	

Special Character	Formal Name	Unicode Code Point	Note
`	Grave Accent	U+0060	
	Space	U+0020	
*	Asterisk	U+002A	
^	Circumflex Accent	U+005E	
)	Right Parenthesis	U+0029	
=	Equals Sign	U+003D	
!	Exclamation Mark	U+0021	
-	Hyphen-Minus	U+002D	
(	Left Parenthesis	U+0028	
+	Plus Sign	U+002B	
~	Tilde	U+007E	
_	Low Line	U+005F	
ä	Latin Small Letter A With Diaeresis	U+00E4	Uppercase letter Ä (U+00C4) is not supported.
ö	Latin Small Letter O With Diaeresis	U+00F6	Uppercase letter Ö (U+00D6) is not supported.
ü	Latin Small Letter U With Diaeresis	U+00FC	Uppercase letter Ü (U+00DC) is not supported.
ß	Latin Small Letter Sharp S	U+00DF	Uppercase letter ẞ (U+1E9E) is not supported.

Some reserved words are not allowed. For details, see 3379404.

Related Information

Data Types and Data Type Conversion
Create and Define a Production Process
Create a Subscription
Create a Timer
Create a Business Rule

Import Production Processes from Other Production Process Design

Instead of creating production processes directly in the production process design, you can also import production processes of the same or different runtime environment from another production process design.

Prerequisites

You have either role of `Automation_Engineer` or `Production_Engineer`. Note that automation engineer can only import automation sequences from other production process designs.

You're assigned to this production process design.

Procedure

1. Open the **Design Production Processes** app.
2. Open or create a production process design.
3. Choose *(Add)*, then choose **Import Existing Process**.
4. In the **Step 1: Select Production Process Design** window, use the search and filter bar to filter the production process design with the relevant version, status and type.
5. In the **Step 2: Select Production Process** window, check the main process or subprocess you want to import.

Note

The automation engineer can only see automation sequences, while the production engineer can see cloud and edge processes. The production engineer cannot see or import automation sequences.

You can view subprocess information in the **Subprocess** column of the main process. You can also edit the description.

Subprocesses are checked only after their main process is checked, not the other way around. If you uncheck the main process, the runtime environment of the subprocess reverts to its previous status. In the case of a subprocess called by more than one main process, changing the runtime environment of one main process also changes the runtime environment of this subprocess and other main processes. Checking or unchecking all processes resets all the changes you've made.

In general, the production process of cloud and edge runtime environments supports cross-runtime import. However, automation sequences do not support importing across runtimes.

The following scenarios are supported:

Original Runtime	New Runtime	Action
cloud	cloud	Keep the runtime environment of the process you're importing as is. If a process with the same name already exists, give it a new name.
cloud	edge	Select an edge web server as your new runtime environment. If a process with the same name already exists, give it a new name. Any cloud subprocesses are converted to their corresponding edge versions. If possible, SAP Digital Manufacturing cloud services are also converted to SAP Digital Manufacturing for edge computing services. Please note, any objects not supported in the edge process become unavailable.
edge	cloud	Select DMC_Cloud web server as your new runtime environment. If a process with the same name already exists, give it a new name. Any edge subprocesses are converted to their corresponding cloud versions. If possible, SAP Digital

Original Runtime	New Runtime	Action
		Manufacturing for edge computing services are also converted to SAP Digital Manufacturing cloud services. Please note, any objects not supported in the cloud process become unavailable.
edge (web server A)	edge (web server B)	Select another edge web server as your new runtime environment. If a process with the same name already exists, give it a new name. All edge subprocesses under web server A are now assigned to the new web server B. Please note, any edge processes previously registered in the service registry under web server A become unavailable.
edge (web server A)	edge (web server A)	Keep the runtime environment of the process you're importing as is. If a process with the same name already exists, give it a new name.
Production Connector (web server A)	Production Connector (web server A)	Keep the runtime environment of the process you're importing as is. If a process with the same name already exists, give it a new name.

6. Choose **Import**.

Results

You can see production processes imported in the service library under the category of "Production Processes". Any change of the imported production process will not affect the original and vice versa.

Create and Use Snippets

You can reuse part of production process across production process designs by creating snippets.

A snippet is a small section of re-usable process components consisting of BPMN event elements and services. Snippets can contain consecutive or inconsecutive services, codes, parameters, and local variables. Snippets don't support subprocesses or start/end elements.

Create a Snippet

1. Open a production process.
2. Use Shift key to select more than one element of the process.
3. Choose *(save)* next to the selection area.
4. In the **Save Snippet** window, input a unique name for your snippet.
5. Choose **Save**.

Use a Snippet

1. Open a production process where you want to add a snippet.
2. Under **Services and Processes**, choose **Select Services**.

   > **Note**
   >
   > You can see snippets created by yourself and other users under the **Snippets** group. The system only displays snippets for the current runtime. When you delete a snippet in the **Manage My Snippets** tab, it isn't deleted from production processes that contain the snippet.

3. Drag and drop a snippet onto the canvas.

Download Production Processes

You can download a production process into an image file (PNG) or a BPMN file (XML).

Prerequisites

You have the role of `Automation_Engineer` or `Production_Engineer`.

Context

To download a process, open the process in the editor and then choose the *(Download)* button.

> **Note**
>
> If a production process embeds one or more other production processes, the downloaded BPMN file or image does not contain the data of the embedded production processes. You can navigate into each embedded production process and download its own BPMN file or image.

Standard Response Codes of Service and Production Process

When calling a service or production process in the Production Process Designer, you may receive different response codes depending on the execution result.

Response Code	Description
400	The server cannot understand the request due to invalid syntax.
401	The request has not been applied because it lacks valid authentication credentials for the production process.
403	The server understood the request but refuses to authorize it due to insufficient permissions to the production process.
404	The server can't find the requested service or production process.
409	The request could not be completed due to a conflict with the current state of the resource.

Response Code	Description
429	The server has received excessive number of requests in a defined time window. You can automatically retry the service after it returns the 429 response code. Or, you can use the error catch element to catch this error code thrown by the service. Then, you can perform further actions at your discretion.
500	Internal error occurred when calling the production process. Possible errors: `errProcessDefinitionNotFound`: Process definition key and version cannot be found in tenant database. `errProcessDocParamsMissReqParam`: Required input parameters are missing. `errProcessDocParamsMissDataType`: Data type information is missing. `errProcessDocParamsUnMatch`: The data type of parameter value does not match the data type defined. For the above cases, contact your production engineer that designed the production process to find out a solution.
502	A server acting as a gateway or proxy received an invalid response from the upstream server.
503	The server is currently unavailable (overloaded or down for maintenance).
504	The synchronous call of the production process timed out.

You can find the list of all response codes related to each production process and the response schema in the **Manage Service Registry** app.

Expose Automation Sequence as RESTful Service

Expose automation sequences as RESTful services from the **Design Production Processes** app.

Prerequisites

- You have activated the web server on Production Connector in the **Configure Production Connectivity** app and configured related settings. Upload the RESTful service provider certificate if you want to call the automation sequence exposed as a RESTful service with HTTPS. This ensures that data is transmitted with encryption. For details, see Enabling a Production Connector as a RESTful Web Server.
- You have the role of `Automation_Engineer`.
- You have created a production process design with an automation sequence of Production Connector on Windows runtime.

Context

Sometimes, the assets or systems on the shop floor don't support calling each other directly with the OPC protocol. In these cases, you can expose automation sequences as RESTful services that can act as a bridge between assets or systems. Once assets

or systems complete operations, they can call back these automation sequences directly to proceed with further actions, such as shop floor services (OPC UA methods).

ℹ Note

We only support the HTTP protocol (HTTP and HTTPS) when calling from assets or systems to automation sequence.

You can use the **Manage Service Registry** app to view these services centrally. Unlike production processes published to the service registry, automation sequences are exposed as RESTful services only to the shop floor local network. For details, see Registering Services in the SaaS Tenant of SAP Digital Manufacturing.

ℹ Note

The automation sequence must run in a Production Connector on Windows version 2.6.0 or higher environment, to be exposed as a RESTful service. Production Connector on edge is not supported.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design with an automation sequence of Production Connector on Windows runtime.
3. Choose **Edit Header** next to the automation sequence name on the detailed page.
4. Switch on the toggle of **Expose as RESTful Service**.

   ### ℹ Note

   The status of **Expose as RESTful Service** in the header of the automation sequence will change to "Yes" once you switch this toggle on, regardless of whether the automation sequence, as a RESTful service, is successfully deployed to the runtime or not.

5. Configure the **Advanced Settings**.
   a. The **URL/Path** field is autogenerated. You can also edit it, but ensure that it does not start or end with "/", and there should be no duplicate paths for automation sequences on the same web server.
   b. Enable at least one protocol (**HTTP** or **HTTPS**). This option gives you the freedom to choose if the endpoint can be accessible via HTTP, HTTPS, or both. HTTP should be avoided and used only when the caller does not support HTTPS. In case the endpoint has HTTP enabled, further authentication or authorization cannot be provided due to security concerns. If the endpoint access type is set to **Authenticated**, then the protocol can only be set to HTTPS to ensure that JWT credentials are transmitted encrypted.
   c. Choose from the dropdown list the access type:
      - **Anonymous**: No authentication method is used in the calling.
      - **Authenticated**: If the endpoint access type is set to **Authenticated**, the caller must have a valid JWT token issued by the central Authorization Service. For details, see 3643190.
6. Choose **Confirm**.

Next Steps

There are still cases where deployment issues happen. You need to restart the agent instances on the Production Connector and redeploy the production process designs.

Once you have deployed and activated the production process design, you will see the automation sequence that you have exposed as RESTful service in the **Manage Service Registry** app under the **Automation Sequence** service group and view its endpoint detail information.

Deploy and Activate a Production Process Design

After completing a production process design, first assign it to a deployment group and then deploy the design from the deployment group to the runtime environment. This ensures that the deployment is in the control of fewer people and the disruption to the production environment can be minimized.

Prerequisites

- You have the role of `Automation_Engineer` or `Production_Engineer`.
- You're assigned to this production process design.
- The production process design is in the status of **Draft**.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. Choose **Select Deployment Group** in the dropdown menu of **Quick Deploy**.

   > **Note**
   >
   > For a fast deployment, you can choose **Quick Deploy** to complete the following steps all at once.
   >
   > To avoid potential downtime when you deploy a modified production process design but don't activate it, while the old design is archived, you are recommended to choose **Quick Deploy** so that both operations are performed at the same time for the modified production process designs. This feature applies only to compatible changes in cloud or edge processes. Automation sequences and subscriptions are not supported.
   >
   > The system will automatically validate if all the conditions of relevant services and indicators are met for the production process design to be deployed. The validation report will show which services or indicators in which production process have issues. You can fix these issues with the information first and choose **Validate** to validate your design again.

4. In the **Add to Deployment Group** window, confirm the default deployment group automatically created and choose **Submit for Deployment**. The status of the production process design becomes **Awaiting Deployment**.

   Alternatively, instead of using the default deployment group, you can select other existing deployment groups by choosing **Select Another Group**. In the **Select Deployment Group** window, you can also choose *(add)* to create a new deployment group. If you create a new deployment group, you become its administrator automatically.

5. In the **Deployment Group** window, adjust the shop floor elements that you want to deploy.
6. Choose **Deploy and Activate**.

   > **Note**
   >
   > You can still go to the **Deploy Shop Floor Elements** app to deploy the production process design afterwards. See Deploy and Activate Shop Floor Elements for details.
   >
   > There are also cases where production process design was activated while some agents of relevant shop floor systems could not be started because remote shop floor systems are not available or servers are down under planned maintenance, for example. You can review the information in the activation exception report in the **Deploy Shop Floor Elements** app and try to restart the impacted agent instances later. To check details or to start the agent instances, choose the shop floor system and navigate to the shop floor system maintenance page.

By default, automation engineer is allowed to submit and deploy production process design that has only Production Connector runtime production process. Otherwise, for production process design with cloud/edge runtime production process or all three runtime production processes, only production engineer is allowed to deploy and submit.

Results

The status of the production process design becomes **Deployed**.

For production process with parameters of complex data type, you can view the details of each parameter by choosing the **Details** button even after its deployment.

Related Information

Deploy Shop Floor Elements
Deploy and Activate Shop Floor Elements

Debug Deployed Production Processes

You can debug a cloud production process after deployment in Production Process Designer.

Debugging helps you find out the problem by checking process execution step by step.

> **Note**
> Debugging supports cloud process only.

Prerequisites

You have the role of `Production_Engineer`.

Context

One process can be only debugged by one user at the same time.

You can start another debug session for the subprocess from design time.

For parallel execution, by default one branch will be selected automatically. Choose **Next Step** to execute the selected step and stop at the next step. You can switch to other branch by selecting the step in other branch.

In the production process debug session, if it takes a long time for a step to proceed, you can see that step highlighted.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. Select a production process and choose **Debug**.
4. If required, enter input values in the debug production process window and choose **Debug**.

5. In graphic view, switch on the **Edit Breakpoint** toggle and click on the step that you want to add breakpoint. To delete all breakpoints at once, choose **Delete All Breakpoints**.
6. Choose **Next Breakpoint** to let the debugging process stop at the next breakpoint. The previous steps are executed.
7. Choose **Next Step** to let the debugging process stop at the next step. The previous step is executed.
8. Choose **Restart Debugging** to terminate the current debugging process and start another one.
9. Choose **Exit Debugging** to end the debug session and go back to the normal design time of Production Process Designer.

ℹ Note

The production process being debugged is displayed separately in the **Monitor Production Processes** app with the status "Running (debugging)". The number of debugging production process is counted under the "Running" column of the KPI carousel. Debugging does not affect the production process in normal execution.

Subprocess is executed automatically when you debug the main process. You can double-click the subprocess to check the detail of it. Alternatively, you can also stop at the subprocess step and choose **Debug Subprocess** if you want to debug the subprocess. This will navigate you to the subprocess and stop at the start event of the subprocess. You can then execute steps one by one or set breakpoints in the subprocess. Choosing **Next Step** on the end event will navigate you back to the main process and stop at the next step after this subprocess. If you choose **Next Breakpoint** but there are no more breakpoints in the subprocess, you will be navigated back to the main process and stop at the next breakpoint after this subprocess.

If you want to have an overview of the main process when debugging the subprocess, choose the back button of the navigation bar. Note that you cannot do any other operations on the main process when debugging subprocess. You can double-click the subprocess to continue the subprocess debugging. Choose **Restart Debugging** will terminate the main process and start another one. Choose **Exit Debugging** will terminate the subprocess as well as the main process and go back to the normal design time of Production Process Designer.

Production Process Debug Page

Graphic View

The graphic view provides a whole picture of the process workflow as you see in normal mode. You have also the option to zoom in, zoom out the graphic, and zoom back to its original size.

The breakpoint is the stop point you can set to control which step the debugging process is going to stop at. You can use the **Edit Breakpoint** toggle to add breakpoint on different elements.

Selected Step View

The step view shows the selected step's name, description, step ID, input and output parameter information.

Processing Step - Input and Previous Step - Output View

The parameter view shows the current processing step name, input parameter name, type, expression and value; the previous step name, output parameter name, type and value. You can edit the previous step output parameter value to update next steps' values.

ℹ Note

Executed step will not execute again even if its input is edited.

Processing Step - Log View

The log view shows real-time log information with debug log level.

Parameters and Variables View

The parameter and variable lists local variables, output parameters of previous steps, input parameters of the process instance. You can edit the value of each parameter.

> **Note**
>
> You can configure which local variable or parameter to display by choosing **Select Variables**.

Related Information

Monitor and Track Production Processes
Configure Debug Mode Settings

Run a Production Process After Deployment

After a production process is deployed and activated, you can run it directly from the cloud/edge and verify if the production process actually works in the runtime environment.

Context

When you create a cloud/edge process, you can choose whether to run it asynchronously or synchronously. By default, cloud/edge processes run synchronously which means that you can see outputs immediately. If you choose to run a cloud/edge process asynchronously, you can only check for the execution status and, if successful, the output in the **Monitor Production Processes** app.

An automation sequence is always run synchronously.

Procedure

1. Open the **Design Production Processes** app.
2. Open a production process design.
3. Select a production process and choose **Run**.
4. If required, enter input values.

   > **Example**
   >
   > - Enter `"red"` for a string value.
   > - Enter `["red", "pink"]` for a string array value.
   > - Enter `100` for an integer value.

5. If required, for a cloud/edge process in the current Production Process Designer, you can choose whether or not to run it asynchronously or synchronously. You can also change the log level of the cloud process in the **Production Process Administration** app and see the log detail in the **Monitor Production Processes** app. For more information, see Log Level and Monitor and Track Production Processes.

   > **Note**

The **Default** log level is the log level you set in the **Production Process Administration** app. If no log level is set, the default option will use the **Error** log level.

6. Choose **Run**.

Note

For synchronously-run production processes, complex data type output parameter value is displayed as a link. You can click the link and choose **Copy** to copy the whole value.