_designer_excerpts

Chapter	Name	Excerpt	Usage
Adapters	**info_alias_attribute_action**	The adapter action derives from the used operation. Do not configure this.	• MongoDB • Memory • Filesystem • Logger • FlatFile • S3 • SQL
Adapters	**supported_databases**	PAS supports the following database client libraries: DB2, Microsoft SQL Server, MySQL, MariaDB, Oracle and SQLite. If you want to use other database systems, please contact the PAS support team.	• Database-Specific Mappings • Database Server-Specific Notes for SQL Adapters
Adapters	**url_adapter_http_headers**	With xUML service adapter calls, the xUML Runtime adds the following outgoing HTTP headers containing correlation information to the request: • **X-Transaction-Id** or **xTransactionId** (in JMS context) This header identifies the transaction the call belongs to. You can set the transaction id manually with setTransactionID. If not set, the Runtime will generate one. This header will be passed through the callstack to identify all service calls that belong to a transaction. • **X-Request-Id** This header identifies the unique request. The Runtime generates a unique number for each adapter call. • **X-Sender-Host** and **X-Sender-Service** These headers contain the sender host resp. the sender service. They are set by the Runtime automatically. Transaction id and request id will be logged to the transaction log on the adapter call (refer to Administration Guide > Contents of the Transaction Log for more details). Having this information, you can use this for error analysis or usage metrics.	• URL Adapter • Using the URL Adapter with the HTTP Protocol
Apache Kafka	**kafka_message**	The Apache Kafka documentation speaks of events that contain a message, or of records in more technical parts of the documentation. This documentation summarizes all under the term "message".	• Apache Kafka Producer Adapter pp.
Configuring the Instance List	**instance_list_basics**	Each application offers two basic functionalities during execution: The user can start a process (and create a so-called **process instance**) and he can view the existing instances that have not yet been completed. These "instances in progress" are displayed in the instance list. For each BPMN model, a dedicated instance list is generated.	• Configuring the Instance List • Configuring the Columns of the Instance List • BPMN Model
Configuring the Instance List Supported Form Elements	**add_column**	Use **Add** to add a new column.	• Configuring the Instance List • Data Table

Configuring the Instance List Supported Form Elements	**column_cancel**	Use **Cancel** to discard your changes.	• Configuring the Instance List • Data Table
Configuring the Instance List Supported Form Elements	**column_drag**	Use this area to move the columns by drag and drop.	• Configuring the Instance List • Data Table
Configuring the Instance List Supported Form Elements	**column_hidden**	Activate this checkbox to hide this column during execution.	• Configuring the Instance List • Data Table
Configuring the Instance List Supported Form Elements	**column_label**	The input in this field is displayed as header of the column.	• Configuring the Instance List • Data Table
Configuring the Instance List Supported Form Elements	**column_name**	Technical identifier for data processing. Not visible for the end user. The input in this field is mandatory.	• Configuring the Instance List • Data Table
Configuring the Instance List Supported Form Elements	**column_save**	Use **Save** to save your changes.	• Configuring the Instance List • Data Table

Constants and Names

service_types_parameters

Protocol	Parameter	Description
TCPIP	RemoteIPAddress	IP address of host that initiated the request. If a proxy is used, this is the IP of the proxy server.
	RemotePort	Port number, from which the request was initiated.
	RemoteHostName	Name of the host that initiated the request. If not resolvable, the IP address is returned.
	LocalIPAddress	IP address of the machine this service runs on.
	LocalPort	Port number of this service.
	LocalHostName	Name of the host this service runs on. If not resolvable, the IP address is returned.
HTTP	Method	HTTP method used to invoke this service. Usually, it returns **POST**.
	URI	URI called to invoke this service.
	Body	Complete content of the HTTP request invoking this service.
	<header>	Any other name will return the value of the HTTP header of the same name, if it exists.
SOAP	Header	Complete source of the SOAP header sent to invoke this service.
	Body	Complete source of the SOAP body sent to invoke this service.
	fault	Content of the SOAP fault if a request comes with a fault element.
System	Host	Local host name. This call returns a value for all service types.
	IP	IP address of the local host. This call returns a value for all service types.
	User	The operating system user who is executing the current service. This user has nothing to do with the external user who might invoke a service. Thus, this call returns a value for all service types.
SystemEnvironment	<environment variable name>	Value of the operating system environment variable. This call returns a value for all service types. Note: the accessible environment variables can be inspected in the **start log** of the current service. Since Linux is case sensitive, you should specify all environment variables as they are.
Object	class	The name of the persistent state class.
	ID	The unique technical ID of the of the object as stored to the persistent state database.

- getServiceContext() Function
- getServiceContextValue() Function

<table>
<tr>
<td>Constants and Names</td>
<td>service_types_protocols</td>
<td>
<table>
<tr><th>Service Type / Context</th><th>Protocols</th></tr>
<tr><td>All</td><td>System
SystemEnvironment</td></tr>
<tr><td>HTTP</td><td>TCPIP
HTTP</td></tr>
<tr><td>SOAP</td><td>TCPIP
HTTP
SOAP</td></tr>
<tr><td>REST</td><td>TCPIP
HTTP</td></tr>
<tr><td>Persistent State</td><td>Object</td></tr>
</table>
</td>
<td>getServiceContext() Function
getServiceContextValue() Function</td>
</tr>
<tr>
<td>Constants and Names</td>
<td>transaction_id</td>
<td>The Transaction ID identifies a transaction. It is a unique number used to trace service calls through the call stack of multiple service calls.
Clients calling an xUML service can provide a transaction ID in HTTP header X-Transaction-ID or xTransactionId (in JMS context).
SOAP clients can also use the SOAP headers to provide a transaction ID.
If an xUML service is called without providing a transaction ID, the Runtime will generate such an ID.
This ID will be passed on through the call stack of the xUML service, so that the whole transaction can be traced. This can be useful, when analyzing the log file in case of error.</td>
<td>getTransactionID() Function
setTransactionID</td>
</tr>
<tr>
<td>Designer Administration</td>
<td>backup_upload_change_file</td>
<td>The name of the selected file is displayed. If you want to select a different file, use option Change File.

The file's name is also displayed in field Name.

If you download a backup file from the Designer administration, the file name is extended by the creation date of the backup (timestamp).</td>
<td>Creating and Restoring Backups of Namespaces</td>
</tr>
<tr>
<td>Designer Administration</td>
<td>backup_upload_change_name</td>
<td>Change the contents of the Name field of you want to rename the backup before uploading it.

Option Confirm is only enabled after you have selected the namespace to which the file is to be uploaded.</td>
<td>Creating and Restoring Backups of Namespaces</td>
</tr>
<tr>
<td>Designer Administration</td>
<td>library_documentation</td>
<td>If documentation is available, it will be opened in a new browser tab.

This allows developers to access library documentation at any time during the development process.</td>
<td>Administrating Libraries
Libraries</td>
</tr>
<tr>
<td>Designer Administration</td>
<td>note_backups_exclude_libraries</td>
<td>Libraries are not included in a namespace backup. If you want to backup a library, use the export functionality in the Controls panel. Refer to Creating a Library for details.</td>
<td>Creating and Restoring Backups of Namespaces (2x)</td>
</tr>
<tr>
<td>Designer Administration

Getting Started With Designer</td>
<td>different_admin_views</td>
<td>The content of the tree in the administration tab depends on the profile assigned to your account within Scheer PAS Administration.</td>
<td>PAS Designer Administration
Working with the Designer</td>
</tr>
</table>

Designer Panel Overview	**note_bpmn_model_export**	A BPMN export does only contain the BPMN model. Execution diagrams are not included.	• Exporting and Importing Service Panel Contents (2x)
Designer Panel Overview	**note_service_panel_name_restrictions**	**Restrictions on Element Names** **A BPMN model name must be unique within one service.** In addition, the following name restrictions apply to all service panel elements: Element names... • ... must not be empty. • ... must not contain spaces. Exception: Spaces are allowed in operation names. • ... must not start with numbers. • ... must not end with a period (**.**). • ... must not contain one of the following characters: **<**, **>**, **:**, **"**, **/**, ****, **|**, **?**, *** • Furthermore, the following strings must not be used as element names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9.	• Service Panel • Process • BPMN Model • Forms • Single Form • API • Implementation • Modeling BPMN • Modeling Forms • Modeling Data Mapping • Modeling Activities
Designer Panel Overview Implementing Your Process	**undo_redo_changes_in_implementation**	In the Implementation folder, you can undo or redo (after undo) your previous changes using the corresponding functions in the Designer editors or using the corresponding keyboard standard shortcuts (Ctrl+Z/Y).	• Implementation • Modeling Data Structures
Designer User Guide	**add_horizontal_space**	Use this option if you need more or less space on the diagram pane. Click on the pane to display the start line, the second click marks the end line and defines the size of the area to be added. To add space, click from left to right to define the area. If you want to delete a horizontal space, click from right to left to define the area	• Working with the BPMN Editor
Designer User Guide	**add_vertical_space**	Use this option if you need more or less space on the diagram pane. Click on the pane to display the start line, the second click marks the end line and defines the size of the area to be added. To add space, click from top to bottom to define the area. If you want to delete a vertical space, click from bottom to top to define the area.	• Working with the BPMN Editor

Designer User Guide	**panel_customization**	You have several options to adjust the diagram pane. When you open the pane for the first time, the default view is displayed:
		
		• The **Diagram Pane** is displayed in the centre at the top (refer to chapter Modeling BPMN for further information)
		• The **Execution Pane** is displayed below the diagram pane (refer to chapter Modeling Execution for further information)
		• Four **Designer Panels** are shown in the left sidebar
		  ◦ The **Service** Panel
		  ◦ The **Controls** Panel
		  ◦ The **Attributes** Panel
		  ◦ The **Validation** Panel
			• Working with the Execution Editor
			• Working with the BPMN Editor

Designer User Guide	**show_invalid**	Activate this option to highlight all invalid elements.	• Working with the BPMN Editor
Designer User Guide	**tip_info_about_roles_in_admin_guide**	Refer to Administration Guide > Managing Roles for detailed information about roles.	• Services • Folders
Designer User Guide	**toggle_grid**	Use this option to enable or disable the grid on the diagram pane. The grid supports you during modeling: If the grid is enabled, elements snap to the grid points.	• Working with the BPMN Editor

<table>
<tr>
<td>Drawing a Business Process</td>
<td>exclusive_gateway_conditions</td>
<td>

If you use an exclusive gateway, only one sequence flow is selected.
If no condition evaluates to true and you have not defined a default flow, the xUML Runtime will throw an exception. We recommend avoiding this scenario.

</td>
<td>

Exclusive Gateway
Modeling Decisions

</td>
</tr>
<tr>
<td>Drawing a Business Process</td>
<td>instance_persistent_state</td>
<td>Process instances are kept in the persistent state database: The instances are created on process start and deleted on process end. Refer to xUML Service State Machines for further information.</td>
<td>

Modeling Process Start
Modeling Process End

</td>
</tr>
<tr>
<td>Filesystem Adapter</td>
<td>hint_backslash_in_path</td>
<td>When using the Windows style with backward slashes "\" you have to be aware that you need to escape this character. The escape character is also the "\".
To avoid this, use forward slashes with Windows as well.</td>
<td>

Filesystem Adapter

pp.
</td>
</tr>
<tr>
<td>Form Elements</td>
<td>prepopulation_no</td>
<td>This element cannot be prepopulated.</td>
<td>

Button
Date Picker
Grouping
Label
Subform

</td>
</tr>
<tr>
<td>Form Elements</td>
<td>prepopulation_yes</td>
<td>This element can be prepopulated.

You can set a default value in the Attributes panel to be applied before displaying the form. Refer to Changing BPMN Element Attributes for more information.
You can overwrite these defaults dynamically in the Get Data execution of a user task. How to do this is explained on Using Forms.

</td>
<td>

Checkbox
Image (Form)
Radio Button Group
Select Field
Text Area
Text Box

</td>
</tr>
<tr>
<td>General</td>
<td>designer_working_steps_documentation</td>
<td>
Guidance for Designer Documentation

Creating and running services in Designer consists of several steps, all of which are explained in detail in the documentation:
<table>
<tr><th>Working Step in the Designer</th><th>Related Documentation</th></tr>
<tr><td>Service Development</td><td>Creating Connectors
Modeling BPMN
Modeling Forms
Modeling APIs
Implementing Your Process
Working With Libraries
Configuring the Instance List</td></tr>
<tr><td>Service Validation and Testing</td><td>Validating and Testing a Service</td></tr>
<tr><td>Service Deployment</td><td>Deploying a Service</td></tr>
<tr><td>Service / Application Execution</td><td>Running Designer Applications</td></tr>
</table>
</td>
<td>

Designer Guide - Home
Getting Started With Designer
Working with the Designer
PAS Designer User Guide
Implementing Your Process
Validating and Testing a Service
Running Designer Applications

</td>
</tr>
</table>

General	**tip_increasing_version_number**	**Expert Advice** By default, each newly created service gets number **0.1.0** assigned. We recommend to increase the version number before redeploying each time you have made relevant changes to the service. You can change the version number in section **General** of the **Deployment Properties**. Follow the concept of semantic versioning. In case of deployment problems, comparing the version number and the number of the deployed version can help to find out which version of the service is running.	• Deployment as Container • Deploying a Service • Developing with Designer from Version 23.1
Implementing Your Process	**confirm_deletion_parameter**	Confirm the security prompt in the pop-up window with **Yes** to delete the selected parameter from the operation.	• Adding Parameters in the Mapping Editor • Adding Parameters in the Activity Editor
Implementing Your Process	**default_value_attribute**	Specify a default value for the related variable here.	• Changing the Attributes of Execution Elements • Persisting Data
Implementing Your Process	**deleting_parameters_service_panel**	You can delete a parameter in the implementation folder of the service panel using the **Delete** option of the context menu or the **Del** key.	• Adding Parameters in the Mapping Editor • Adding Parameters in the Activity Editor

Implementing Your Process	**execution_preliminaries**	You need to perform two steps to implement executional parts to your model:
		
		1. Provide all necessary data types and operations for the implementation of your process. These types and operations reside in the **Service** panel of the BPMN editor.
		  • You can use the **Base Types** that are provided with the Designer.
		  • You can create other necessary types yourself in the **Implementation** section. Refer to Modeling Data Mapping for further information.
		  • You can import a library that provides additional types and operations. Refer to Designer Administration > Libraries for further information.
		2. In the second step, select data types and operations from the **Service** panel, and add them to your process at the right places.
		  • How this is done will be explained in this chapter.
			• Modeling Execution
			• Working with the Execution Editor
			• Adding Variables
			• Adding Operation Calls

Implementing Your Process	**execution_tabs**	• **On Event:** The model is executed when the event occurs. • **On Exit:** The model is executed after completion of the task/event. • **Decision:** The model is executed when the process reached the gateway. • **Get Data:** The model is executed when BPMN is waiting in a user task and returns the default values for the form.	• Modeling Execution • Working with the Execution Editor

Implementing Your Process	**info_add_parameters**	You can add parameters only to custom operations, **not to operations of Connectors** and **Libraries**.	• Adding Parameters in the Mapping Editor • Adding Parameters in the Activity Editor • Adding Parameters in the Execution Editor
Implementing Your Process	**local**	Variables created in section **Local** are only available for use within the current execution.	• Modeling Execution • Adding Variables • Adding Operation Calls
Implementing Your Process	**persisted**	Variables created in section **Persisted** are usable in all executions of the BPMN model.	• Modeling Execution • Adding Variables
Implementing Your Process	**persisted_variable_added**	The variable is added to the execution pane.	• Adding Variables • Persisting Data
Implementing Your Process	**second_persisted_variable**	Once a variable is defined as to be **persisted**, it is available in all executions throughout the BPMN model. If you drag out the same type again, a second persisted variable is created that is independent of the first. A consecutive number is appended to the name automatically as variable names must be unique.	• Adding Variables • Persisting Data
Implementing Your Process Modeling APIs	**add_action_script**	Add an action script operation to the class.	• Modeling APIs • Modeling Data Structures
Implementing Your Process Modeling APIs	**add_activity_operation**	Add an activity operation to the class.	• Modeling APIs • Modeling Data Structures
Implementing Your Process Modeling APIs	**add_interface**	Add an interface to the class.	• Modeling APIs • Modeling Data Structures
Implementing Your Process Modeling APIs	**add_mapping_operation**	Add a mapping operation to the class.	• Modeling APIs • Modeling Data Structures

<table>
<tr><td>Implementing Your Process

Modeling APIs</td><td>add_operation</td><td>Add an operation to the class. The implementation is to be defined later.</td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>add_operation_cm</td><td>Add an operation to the class.</td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>add_sub_class</td><td>Add a sub-class to the class.</td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>copy_class</td><td>Copy the class to paste it elsewhere to the API or Implementation folder.</td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>cut_class</td><td>Cut the class to paste it elsewhere to the API or Implementation folder.</td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>delete_class</td><td>Delete the class.</td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>interface_context_menu</td><td><table><tr><th>Menu Item</th><th>Description</th></tr><tr><td>Add Class</td><td>Add a class or sub-class to the interface. Classes within interfaces can be nested.</td></tr><tr><td>Add Interface</td><td>Add another interface to the interface. Interfaces can be nested.</td></tr><tr><td>Add Operation</td><td>Add an operation to the interface.

Operations of interfaces do not have an implementation but only define the signature (parameters and types).</td></tr><tr><td>Cut</td><td>Cut the interface to paste it elsewhere to the API or Implementation folder.</td></tr><tr><td>Copy</td><td>Copy the interface to paste it elsewhere to the API or Implementation folder.</td></tr><tr><td>Paste</td><td>Paste the interface elsewhere to the API or Implementation folder. Available if Copy or Cut option have been used before.</td></tr><tr><td>Rename</td><td>Change the name of the interface.</td></tr><tr><td>Delete</td><td>Delete the interface.</td></tr></table></td><td>Modeling APIs
Modeling Data Structures</td></tr>
<tr><td>Implementing Your Process

Modeling APIs</td><td>interface_quick_action</td><td><table><tr><th>Quick Action</th><th>Description</th></tr><tr><td>  </td><td>Add an operation to the interface. Operations of interfaces have no implementation, they only describe the signature of the operation.</td></tr></table></td><td>Modeling APIs
Modeling Data Structures</td></tr>
</table>

<table>
<tr>
<td>Implementing Your Process

Modeling APIs</td>
<td>operation</td>
<td>An operation adds behavior to a class or interface. The behavior describes how to process the data given by the parameters. In the context of the Designer, you can implement operations as mapping, action script or activity.</td>
<td>Concepts of Data Modeling
Modeling APIs</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling APIs</td>
<td>operation_context_menu</td>
<td>
<table>
<tr><th>Menu Item</th><th>Description</th></tr>
<tr><td>Add Parameter in</td><td>Add an input parameter to the operation.</td></tr>
<tr><td>Add Parameter out</td><td>Add an output parameter to the operation.</td></tr>
<tr><td>Add Parameter return</td><td>Add a return parameter to the operation.</td></tr>
<tr><td>Add Operation (Suboperation)</td><td>Add a suboperation to the operation.</td></tr>
<tr><td>Open (Implementation)</td><td>Open the implementation of the selected operation in a separate tab.

Available if the operation has an implementation, yet.</td></tr>
<tr><td>Change (Implementation)</td><td>Change the type of implementation or remove the implementation.

Available if the operation has an implementation, yet.</td></tr>
<tr><td>Create (Implementation)</td><td>You can choose between three different types of implementation for your class operations:
Mapping Diagram: Refer to Modeling Data Mapping for detailed information.
Action Script: Refer to Using Action Script for detailed information.
Activity Diagram: Refer to Modeling Activities for detailed information.
Available if the operation has no implementation, yet.</td></tr>
<tr><td>Cut</td><td>Cut the operation to paste it elsewhere to the API or Implementation folder.</td></tr>
<tr><td>Copy</td><td>Copy the operation to paste it elsewhere to the API or Implementation folder.</td></tr>
<tr><td>Paste</td><td>Paste the operation elsewhere to the API or Implementation folder. Available if Copy or Cut option have been used before.</td></tr>
<tr><td>Rename</td><td>Change the name of the operation.</td></tr>
<tr><td>Delete</td><td>Delete the operation.</td></tr>
</table>
</td>
<td>Modeling APIs
Modeling Data Structures</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling APIs</td>
<td>operation_quick_actions

Check if anchor link can be kept - if so, add info box from _designer_excerpts_big here as well</td>
<td>
<table>
<tr><th>Quick Action</th><th>Description</th></tr>
<tr><td>  </td><td>Add an input parameter to the operation.</td></tr>
<tr><td>  </td><td>Add an output parameter to the operation.</td></tr>
<tr><td>  </td><td>Add a return parameter to the operation.</td></tr>
<tr><td>  </td><td>Open the implementation of the operation in a separate tab.

If you have not yet selected an implementation, a dialog opens first, which allows you to select the desired implementation. Refer to Create Implementation for more information.</td></tr>
</table>
</td>
<td>Modeling APIs
Modeling Data Structures</td>
</tr>
</table>

<table>
<tr>
<td>Implementing Your Process

Modeling APIs</td>
<td>parameter_context_menu</td>
<td>
<table>
<tr><th>Menu Item</th><th>Description</th></tr>
<tr><td>Move up</td><td rowspan="2">Change the order of parameters.</td></tr>
<tr><td>Move down</td></tr>
<tr><td>Cut</td><td>Cut the parameter to paste it elsewhere to the API or Implementation folder.</td></tr>
<tr><td>Copy</td><td>Copy the parameter to paste it elsewhere to the API or Implementation folder.</td></tr>
<tr><td>Paste</td><td>Paste the parameter elsewhere to the API or Implementation folder. Available if Copy or Cut option have been used before.</td></tr>
<tr><td>Rename</td><td>Change the name of the parameter.</td></tr>
<tr><td>Delete</td><td>Delete the parameter.</td></tr>
</table>
</td>
<td>

Modeling APIs
Modeling Data Structures

</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling APIs</td>
<td>paste_class</td>
<td>Paste the class elsewhere to the API or Implementation folder. Available if Copy or Cut option have been used before.</td>
<td>

Modeling APIs
Modeling Data Structures

</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling APIs</td>
<td>rename_class</td>
<td>Change the name of the class.</td>
<td>

Modeling APIs
Modeling Data Structures

</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling BPMN</td>
<td>class</td>
<td>A class is an aggregation of properties and operations that describes a complex data type from which objects can be created.</td>
<td>

Modeling Data Structures
Concepts of Data Modeling
Modeling APIs

</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling BPMN</td>
<td>implementation_folder</td>
<td>All elements inside the Implementation folder need to be created within packages. Add a new package via the quick action or the context menu.</td>
<td>

Implementation
Modeling Data Structures

</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling BPMN</td>
<td>packages</td>
<td>A package is like a directory for the file system. It is used to group executable data model elements. Packages can have any depth of nesting: To structure your work, you can create packages within packages.
Also, packages define a sort of namespace to the contained elements. The name of the package is part of the element path, e.g. Package1.Class is different from Package2.Class.</td>
<td>

Modeling Data Structures
Implementation

</td>
</tr>
<tr>
<td>Implementing Your Process

Modeling BPMN</td>
<td>property</td>
<td>Properties are data fields that describe the structure of the class.</td>
<td>

Modeling Data Structures
Concepts of Data Modeling

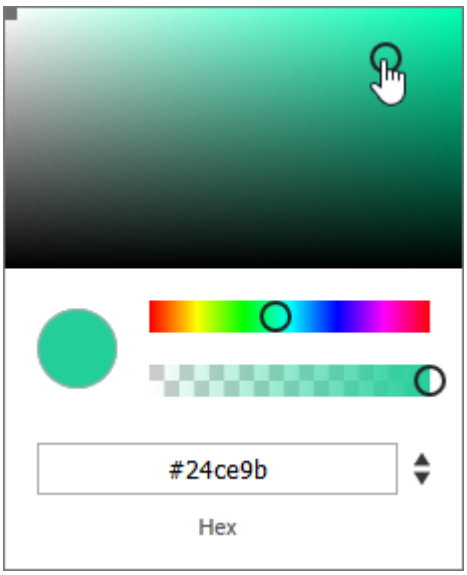
</td>
</tr>
</table>

Implementing Your Process Modeling BPMN Modeling Forms	**undo_last_action**	Undo the previous action you performed in... • ... one of the Designer editors (except mapping editor). • ... the **Implementation** folder of the service panel.	• Working with the BPMN Editor • Working with the Form Editor • Working with the Activity Editor • Working with the Action Script Editor • Working with the JavaScript Editor
Implementing Your Process Modeling BPMN Modeling Forms	**redo_last_action**	Redo your previous action (after undo).	• Working with the BPMN Editor • Working with the Form Editor • Working with the Activity Editor • Working with the Action Script Editor • Working with the JavaScript Editor
Implementing Your Process Modeling Data Structures	**hint_naming_conventions** ⚠ PAS DOC-1658 - Jira project doesn't exist or you don't have permission to view it.	Apply the same naming conventions to all your models. This makes reading a model much easier. Refer to Naming Conventions and Containment Tree Organization in the Builder User Guide for an overview on practice-approved naming conventions.	• Modeling Data Structures • Implementation • Modeling APIs
Implementing Your Process Modeling Data Structures	**interface**	In contrast to a class, an interface has no properties nor implementations. Interfaces are used to define common operations of multiple classes, and then derive from that interface. Operations of interfaces do not have an implementation but only define the signature (parameters and types).	• Modeling Data Structures • Concepts of Data Modeling • Modeling APIs

Implementing Your Process Modeling Data Structures	**parameter**	Operations can have parameters that define the input and output objects. Operation parameters can be of simple type (Base Types) or of complex type (class or interface).	• Modeling Data Structures • Concepts of Data Modeling
Implementing Your Process Modeling Activities	**literal_supported_types**	Literals can only be connected with output parameters, return parameters and operation input pins of base type **Boolean**, **DateTime**, **Float**, **Integer** and **String**.	• Literal • Adding Literals
Managing the Service Details	**enable_angular_build**	Activate this option (**Yes**) to enable the possibility to export your service as an Angular project for further pro-code development. You can find the export option in the additional menu of the controls panel. If the option is enabled, an Angular project will be created for all forms and for the instance table. Deactivate (**No**) the checkbox to disable the export. If the Angular build is disabled, it also will not build the Angular project on deployment. Default is **Yes**. **Expert Advice** If your process does not contain user tasks with assigned forms and if you do not need the instance table, we recommend to disable this option. The deployment will be much faster and the repository file much smaller.	• Deployment to the Integration Component Bridge • Deployment as Container
Managing the Service Details	**enable_validation**	Activate this option (**Yes**) to enable automatic compilation with each change in the respective model. Deactivate (**No**) the checkbox to disable the automatic compilation. If you chose this option, you can trigger the compilation manually via **Start validation** in the controls panel. The corresponding service is also compiled during a deployment, an export of the repository or similar actions, even if automatic compilation is disabled. Default is **Yes**.	• Deployment to the Integration Component Bridge • Deployment as Container
Managing the Service Details	**info_semantic_versioning**	The version of the service must follow Semantic Versioning 2.0.0: **Major.Minor.Patch**. A pre-release tag is optional. A version numbering that deviates from this will lead to errors during deployment.	• Deployment to the Integration Component Bridge • Deployment as Container
Modeling Activities	**hint_activity_diagram**	Activities are modeled in activity diagrams. Refer to Modeling Activities for more information on how to create an activity diagram.	• Adding Operations in the Activity Editor • Drawing the Control Flow • Adding Local Variables • Drawing Object Flow • Moving Pins
Modeling Activities	**info_get_attribute_limitations**	**Limitations of Attribute Get** Using the **Get** attribute it is only possible to access properties of complex types. **The following is not supported:** • Multiple transformation flows to the same input **pin**. • Adding Action Script statements to the **Get** attribute. • Accessing single array elements, e.g. with **myArray[0]**.	• Transforming Data on an Object Flow • Changing the Attributes of Elements on the Activity Diagram

Modeling Activities Drawing an Activity Diagram	**description_guard**	Control flows that are starting from a decision node need to have a guard expression. A guard expression is an expression that evaluates to true or false, and specifies which control flow branch to follow from the decision node on. One of the guard expressions must be `else` to define the branch to follow when none of the guard expressions are true.	• Changing the Attributes of Elements on the Activity Diagram • Control Flow
Modeling Activities Drawing an Activity Diagram	**description_order**	If a decision node branches the control flow into multiple branches, **order** defines the order in which the guard expressions (see **Guard** above) should be evaluated. This is necessary in case multiple guard expressions evaluate to true. The `else` expression does not need to have an order.	• Changing the Attributes of Elements on the Activity Diagram • Control Flow
Modeling APIs	**api**	An API is the entrance port of your service. It can be used to communicate with the service from the outside. There are different kinds of APIs (e.g. REST API, SOAP API, ...) that describe different communication standards.	• Modeling APIs (2x)
Technical Concepts Constants and Names	**http_headers_adapter**	With xUML service adapter calls, the xUML Runtime adds the following outgoing HTTP headers containing correlation information to the request: • **X-Transaction-Id** or **xTransactionId** (in JMS context) This header identifies the transaction the call belongs to. You can set the transaction id manually with setTransactionID. If not set, the Runtime will generate one. This header will be passed through the callstack to identify all service calls that belong to a transaction. • **X-Request-Id** This header identifies the unique request. The Runtime generates a unique number for each adapter call. • **X-Sender-Host** and **X-Sender-Service** These headers contain the sender host resp. the sender service. They are set by the Runtime automatically. Transaction id and request id will be logged to the transaction log on the adapter call. Having this information, you can use this for error analysis or usage metrics.	Technical Concepts • Distributed Tracing Constants and Names • HTTP Header Support
Modeling APIs Technical Concepts	**http_headers**	xUML services read the following incoming HTTP headers containing correlation information: • **X-Transaction-Id** or **xTransactionId** (in JMS context) This header identifies the transaction the call belongs to. You can set the transaction id manually with setTransactionID. This header will be passed through the callstack to identify all service calls that belong to a transaction. • **X-Request-Id** This header should identify the unique request. • **X-Sender-Host** and **X-Sender-Service** These headers should contain the sender host resp. the sender service. These headers will be all logged to the transaction log. Having this information, you can use this for error analysis or usage metrics.	Modeling APIs • REST API • Calling the REST API • HTTP Header Support Technical Concepts • Distributed Tracing
Modeling APIs	**httpHeaderMap**	Header information as a map. The map contains arrays of header value strings whereas the header name is the key of the map. • Header names are lowercase and treated case insensitive. • Multiple headers with the same name are treated as arrays. Refer to HTTP Header Support for more information on the standard xUML HTTP headers.	• REST
Modeling APIs	**maxrequestbodysize**	Specifies the maximum size of the request in KB (1 KB = 1024 Bytes). This can be used to prevent DoS or similar attacks. When the payload of the service exceeds the given maximum, incoming request are rejected.	• REST
Modeling APIs	**maxrequestheadersize**	Specifies the maximum size of the request header in KB (1 KB = 1024 Bytes). This can be used to prevent DoS or similar attacks. When the header payload of the service exceeds the given maximum, incoming request are rejected.	• REST

Modeling APIs	**port**	Ports are elements of the UML modeling standard. A port defines an entrance point to the service and connects the API to an interface or class.	• Modeling APIs (2x)
Modeling BPMN	**activating_lanes**	To activate the lane element, click its border or icon/header. If the element is active, an indicator is displayed in the lower right corner.	• Lane • Modeling Roles
Modeling BPMN	**add_bpmn_elements**	Drag the element you want to create from the toolbar and drop it on the diagram pane. Another way to create a bpmn element is to click on the element in the toolbar, switch to the editor and click again. To cancel the operation, press the **Esc** key.	• Adding BPMN Elements • Working with the BPMN Editor
Modeling BPMN	**add_bpmn_model**	In the **Process** folder you can create your BPMN models. One service can contain several BPMN models. To create a model, hover over the **Process** folder to display the quick start icons and click **Add Model**.	• Modeling BPMN • Process
Modeling BPMN	**add_library**	To add an unused library to the project, click **Add library** .	• Libraries • Adding Libraries
Modeling BPMN	**bpmn_created**	The new BPMN model is also displayed in the service panel and you can use option **Open model** to open it.	• Process • BPMN Model • Modeling BPMN
Modeling BPMN	**bpmn_new_tab**	When you create a new model, it opens automatically in a new tab and you can directly start modeling. Refer to Working with the BPMN Editor for detailed information.	• Process • BPMN Model • Modeling BPMN
Modeling BPMN	**create_bpmn**	Enter a name for the BPMN model in the pop-up window **Create New Model**. Click **Save**.	• Process • BPMN Model • Modeling BPMN
Modeling BPMN	**no_role_assigned_to_lane**	If no roles are assigned to a lane, the process steps inside the lane can be executed by all users.	• Role-based Authorization Concept • Modeling Roles

Modeling BPMN	**other_background_color**	**Background Color** If you want to select a background color, activate the checkbox and click on the color placeholder that will appear. Then use the color picker to select a new color or enter the hexadecimal code of the desired color: 	• Styling BPMN Elements
Modeling BPMN	**tip_export_import_in_service_panel**	Refer to Exporting and Importing Service Panel Contents for further information about the possible options.	• Process • BPMN Model • Forms • Single Form
Modeling BPMN	**tip_service_panel_options**	For detailed explanations of the item options in the **Service** panel, such as cut, copy, rename, and delete, refer to Service Panel.	• Process • BPMN Model • Forms • Single Form • Implementation
Modeling BPMN Implementing Your Process	**attribute_name**	Click here to insert or change the **Name** of the related element. Generated elements (like e.g. **message**) cannot be renamed.	• Changing BPMN Element Attributes • Changing the Attributes of Execution Elements
Modeling BPMN Implementing Your Process	**create_form**	Enter a name for the form in the pop-up window **Create New Form**. Click **Save**.	• Forms • Single Form • Modeling Forms
Modeling BPMN Modeling Forms	**form_created**	The new form is also displayed in the service panel and you can use option **Open model** to open it.	• Forms • Single Form • Modeling Forms

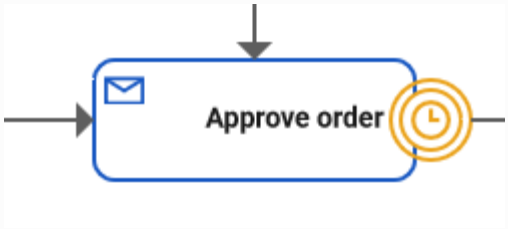
Modeling BPMN Modeling Forms	**form_new_tab**	When you create a new form, it opens automatically in a new tab and you can directly start designing. Refer to Working with the Form Editor for detailed information.	• Forms • Single Form • Modeling Forms
Modeling BPMN Modeling Forms Implementing Your Process	**effect_undo_on_implementation**	If you click **Undo** in one of the Designer editors, this may also have an effect on the **Implementation** folder in the Service panel if you performed your previous action there.	• Working with the BPMN Editor • Working with the Form Editor • Working with the Activity Editor • Working with the Action Script Editor • Working with the JavaScript Editor
Modeling BPMN Supported BPMN Elements	**define_timer**	To define the timer settings, click  to open the timer editor. For detailed explanations see **How to Use the Scheduler/Timer**.	• Changing BPMN Element Attributes • Timer Start Event
Modeling BPMN Supported BPMN Elements	**set_timeout**	To set the duration of the timer, the event should be trigged by a persisted property of type **integer** or **datetime**. You have two options to set the timeout in the attributes panel: • **Integer:** Define the number of seconds the event has to wait. • **Datetime:** Define the absolute datetime when the event should fire.	• Changing BPMN Element Attributes • Timer Event
Modeling BPMN Working With Libaries	**show_library_doc**	Use option **Documentation**  to display the documentation of a library. You can find it in the **Manage Libraries** dialog...	• Libraries • Adding Libraries
Modeling BPMN Working With Libaries	**show_library_doc_in_panel**	... as well as in the service panel when you hover over a library.	• Libraries • Adding Libraries
Modeling BPMN Working With Libraries	**add_libraries_to_folder**	If you want to add your own libraries to a service, hover over the **Libraries** folder in the service panel and use option **Manage Libraries**  .	• Libraries • Adding Libraries
Modeling BPMN Working With Libraries	**library_in_service_panel**	The added library is now displayed in the service panel and you can use it during modeling.	• Libraries • Adding Libraries
Modeling BPMN Working With Libraries	**manage_libraries**	The dialog **Manage Libraries** opens. All libraries that have been uploaded to the current namespace are shown in this dialog: • Currently unused libraries are displayed on the left side of the window. • The libraries that are already used in this service are shown on the right side. Expand the drop-down lists to display the details of each library such as versions and dependencies.	• Libraries • Adding Libraries

Modeling BPMN Working With Libraries	**used_libraries**	The library is now displayed in the column **Used Libraries**. If you have finished, click **Save** to persist your changes.	• Libraries • Adding Libraries
Pro-Code Development	**tip_multi_project_workspaces**	**Expert Advice** For detailed information about multi-project workspaces, visit the Angular documentation.	• Developing Custom Forms in a Library (2x)
Running Designer Applications	**create_new_instance**	Select the process start you want to trigger from the drop-down menu.	• Running Designer Applications • Creating a Process Instance

Running Designer Applications	**info_service_execution_options**	With PAS 23.1, you have two options to execute your service. You can run your applications:
		
		• **in the test service without deployment:** This option enables developers to easily test applied changes. Deployment is not necessary any more to execute the test service. Refer to Working with the Test Environment for details.
		• **in the deployed service:** Developers only need to deploy a service...
		  ◦ if they want to give other team members the opportunity to test dedicated features or versions of the service
		  ◦ if they want to run regeression tests against the test server.
		  ◦ Refer to Deploying a Service for details.
		
		Deployment is also required when your service goes live and is transferred from your test to your productive systen. Refer to Going Live with a Designer Service in PAS Academy > Designer Best Practices for further explanation.
			• Running Designer Applications
			• Creating a Process Instance

Running Designer Applications	**instance_list_content**	The instance list shows all running instances that have not yet been processed to the last service step.	• Running Designer Applications • Creating a Process Instance • Showing the Instance List
Running Designer Applications	**instance_list_progress_view**	A progress view is displayed... • ... if an open process instance is in a state where no forms have to be filled. • ... if the instance is in a state that your role is not allowed to see. You can wait until the process reaches the next form (or the process end) or switch back to the overview and proceed with another instance.	• Creating a Process Instance • Showing the Instance List
Running Designer Applications	**instance_list_refresh**	Use option **Refresh** to update the instance list.	• Showing the Instance List • Running Designer Applications
Running Designer Applications	**instance_list_role_concept**	If the view of the instance list is associated with a role-based authorization concept, only users with appropriate permissions can access it via option that is displayed next to the corresponding process name.	• Running Designer Applications • Creating a Process Instance • Showing the Instance List

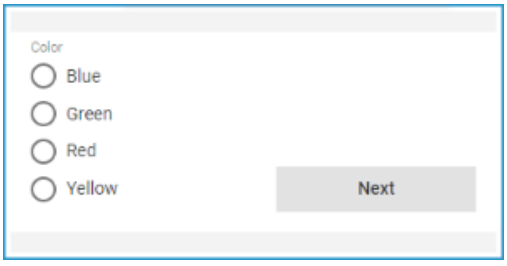
Running Designer Applications	**instance_list_role_label**	If the process is role-based, role labels are displayed - one for each role assigned to your user. Click a label to view only the instances allowed for that role. For detailed information about role usage in apps, refer to chapter Role-based Authorization Concept.	• Running Designer Applications • Showing the Instance List
Running Designer Applications	**search_expressions**	**Search Expressions** • The percent sign (**%**) represents zero, one, or multiple characters. • The underscore sign (_) represents one, single character. To search for a substring, use the percent sign **%** as a wildcard for any character and underscore _ for a single arbitrary character.	• Showing the Instance List (several times)
Running Designer Applications	**select_process_to_create_instance**	Select the process for which you want to create a new instance by clicking the corresponding + option.	• Showing the Instance List
Searching in the Designer Modeling BPMN Working With Libraries Implementing Your Process Validating a Service	**tip_panel_management**	Refer to Customizing Editors and Panels for detailed information about panel management in general.	• Working with the BPMN Editor • Searching in the Designer • Changing BPMN Element Attributes • Styling BPMN Elements • Working With Libraries • Changing the Attributes of Execution Elements • Validating and Testing a Service
Searching in the Designer Working With the Explorer	**explorer_search**	On top of the explorer tree you can find a search box. Select the part of the explorer tree (namespace, folder or service) you want to search, and enter a search term.	• Searching in the Designer • Working With the Explorer
Service Panel	**add_libraries_to_folder_cm**	Alternatively, you can also add libraries via the context menu.	• Libraries
Service Panel Modeling BPMN	**add_model_via_context_menu**	Alternatively, you can open the context menu of a **Process** folder and select the option **Add Model**.	• Process • BPMN Model • Modeling BPMN
Service Panel Modeling Forms	**add_form_via_context_menu**	Alternatively, you can open the context menu of a **Forms** folder and select the option **Add Form**.	• Forms • Single Form • Modeling Forms

Sharing Designer Content	**include_in_library**	The **Include** checkbox corresponds to the element attribute **Include in Library Export**: Activate the checkbox for all elements that you want to include in your library.	• Publishing Assets • Creating a Library
Supported BPMN Elements	**attribute_trigger_events**	**Trigger Events:** Use the attribute **Trigger Events** in the **Attributes Panel** to define which button of a form associated to the element triggers the default process flow of a BPMN process. This attribute is only available for boundary events.	• Plain Event • Message Event
Supported BPMN Elements	**boundary_events**	Plain Event, Message Event and Timer Event can be used as **boundary** events along with User Task and Receive Task. When using the events as boundary events, attach the element directly to the border of the corresponding task: 	• Plain Event • Message Event • Timer Event
Supported BPMN Elements	**creation_of_start_events**	A start event can only be created via the elements toolbar.	• Message Start Event • Start Event • Timer Start Event
Supported BPMN Elements	**editing_and_styling**	• Refer to Working with the BPMN Editor for further information regarding editing of BPMN elements using the different context menus on the diagram pane. • Refer to Styling BPMN Elements for further information regarding styling possibilities for BPMN elements, for example how to change the background color, the font style and size etc.	• **BPMN Elements Template Page** • End Event • Exclusive Gateway • Free Text • Image+ (BPMN) • Plain Event • Message Event • Message Start Event • Parallel Gateway • Receive Task • Relation • Service Task • Start Event • Timer Start Event • Timer Event • User Task
Supported BPMN Elements	**execution_other_elements**	None. The content of this element is ignored during execution.	• **BPMN Elements Template Page** • Free Text • Image+ (BPMN)

Supported Form Elements	**form_elements_cssClass**	Enables a field-acurate layout customization.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Data Table • File Upload • Grouping • Image (Form) • Label • Radio Button Group • Select Field • Subform • Text Area • Text Box • Configuring the Instance List
Supported Form Elements	**form_elements_cssClass_value**	A valid CSS class.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Data Table • File Upload • Grouping • Image (Form) • Label • Radio Button Group • Select Field • Subform • Text Area • Text Box
Supported Form Elements	**form_elements_customer Attributes**	The attribute is used to activate Angular directives, that are created as development kit (devkit) library to expand the default functionality. For detailed information about Angular directives, visit the official Angular documentation. For details about the usage of the **Custom Attributes**, refer to Developing Custom Directives.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Data Table • File Upload • Grouping • Image (Form) • Label • Radio Button Group • Select Field • Subform • Text Area • Text Box

Supported Form Elements	**form_elements_customerAttributes_values**	• empty attribute • attribute with value (e.g. color="red") • multiple entries separated by space (e.g. color="red" multiline max="5")	• **Form Elements Template Page** • Button • Checkbox • Date Picker • File Upload • Grouping • Image (Form) • Label • Radio Button Group • Select Field • Subform • Text Area • Text Box
Supported Form Elements	**form_elements_description**	A commentary field for the developer. The content is invisible to users and its sole purpose is internal documentation.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Label • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_editing**	Refer to Modeling Forms for further information regarding editing of form elements using the context menu.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Label • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_label**	Field name shown in the form.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Label • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_mandatory**	The element must be filled or used if the checkbox is activated.	• **Form Elements Template Page** • Checkbox • Date Picker • Radio Button Group • Select Field • Text Area • Text Box

Supported Form Elements	**form_elements_mandatory_false**	Element may remain empty (default).	• **Form Elements Template Page** • Checkbox • Date Picker • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_mandatory_true**	Element must be filled.	• **Form Elements Template Page** • Checkbox • Date Picker • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_name**	Technical identifier for data processing and not visible for the end user.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Label • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_name_value**	Alphanumercial characters and underscore. It is not allowed to start the name with a number.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Label • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_read-only**	This field is write-protected. When you use the option, **Read-only** is added to the bottom right of the element: 	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Radio Button Group • Select Field • Text Area • Text Box

Supported Form Elements	**form_elements_read-only_false**	User can enter values (default).	• **Form Elements Template Page** • **Form Elements Template Page** • Button • Checkbox • Date Picker • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_read-only_true**	User cannot enter values.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_symbolType**	Displays the type of the form element in read-only mode.	• **Form Elements Template Page** • Button • Checkbox • Date Picker • Label • Radio Button Group • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_text**	In this field, you can enter text. Content added to attribute **Text** is shown in the element. If the user does not change the content of the field during form execution, the content of **Text** will be saved to the database.	• Text Area • Text Box
Supported Form Elements	**form_elements_vertical_alignment**	The attribute is used to adjust the position of a form element in context to a bigger neighboring element. **Example:** The **Radio Button Group** Color is much bigger than the **Button** Next, but the button should be shown at the bottom of the form. Therefore, the button's attribute **Vertical Alignment** is set to **Bottom**. 	• Button • Checkbox • Date Picker • Label • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_vertical_alignment_bottom**	Align the element at the bottom of the element cell.	• Button • Checkbox • Date Picker • Label • Select Field • Text Area • Text Box

Supported Form Elements	**form_elements_vertical_alignment_center**	Align the element in the middle of the element cell.	• Button • Checkbox • Date Picker • Label • Select Field • Text Area • Text Box
Supported Form Elements	**form_elements_vertical_alignment_top**	Align the element on top of the element cell (default).	• Button • Checkbox • Date Picker • Label • Select Field • Text Area • Text Box
Supported Form Elements	**grouping_deletion**	**Deletion Behavior of Grouping Elements** If you delete a grouping element, it is erased from your work area including all content.	• Grouping • Grouping Form Elements
Supported Form Elements	**note_on_instance_table**	The **Instance List** is a special form of the data table which is created for every BPMN model. Refer to Configuring the Instance List for detailed information.	• Data Table • Data Table Handling
Supported Form Elements Modeling Forms	**form_elements_placeholder**	The content of this attribute is displayed in the element when attribute **Text** has not been filled. The placeholder is a sample value, it is not saved to the database. When a user overwrites the placeholder, it disappears and the user's input is saved to attribute **Text**. If content is added to both attributes **Text** and **Placeholder**, the content of **Text** is displayed in the element.	• Configuring Form Elements • Text Area • Text Box
Testing and Integration	**tip_api_organizations**	For detailed information about the usage of organizations in API Management, refer to Organizations in the API Management guide.	• Testing and Integration (2x)
Tips	**click_to_see_gif**	Click the image to see an animated version.	• Various pages (also in ACADEMY space)
Tips	**click_to_see_gif_above**	Click the image above to see an animated version.	• Various pages (also in ACADEMY space)
Validating and Testing a Service	**validating_analyzing_validation_results**	The validation panel is available in all editors, and shows xUML Compiler messages (and) no matter where they occur. Use the information given in the validation panel to check the mentioned elements of your service model.	• Using the Validation • Validation Panel
Validating and Testing a Service	**validating_intro**	The Designer supports you during modeling by validating the service, displaying messages for invalid implementations, and giving you advice on how to fix these errors.	• Using the Validation • Validation Panel

Working With Libraries	**bridge_base**	The Designer provides all necessary base types in a **Bridge Base** standard library. This library is available in all services and cannot be removed. It contains the following xUML base types: • **Any** • **Blob** • **Boolean** • **DateTime** • **Float** • **Integer** • **String** Most of these base types are only able to hold one single piece of information, like text in a string, true or false in a boolean, or binary data in a blob. Refer to Available Base Types for more information on the xUML base types. If you want to associate several bits of information, you have to define a complex type that combines a number of independent base types and possibly other complex types. Such complex types are modeled as classes. To use your own types, you can • define your own data structures in the **Implementation** folder • provide them via a library. How to upload your own libraries is explained on Designer Administration > Libraries.	• Working With Libraries • Adding Variables
Working With Libraries	**library_concepts**	Libraries are code repositories that are useful to organize your development project into re-usable pieces that can be used in multiple services. They contain predefined classes, interfaces, operations and parameters you can use during modeling by simple drag & drop. Libraries are developed with the Builder. The Designer comes with a standard library which already provides all necessary **Base Types** and base type operations.	• Working With Libraries • Concepts of Data Modeling
Working With Libraries	**note_library_deletion**	Be careful when using this option: If you delete a library in the administration, it is no longer available for all services in the namespace. The implementation of the service(s) in the affected namespace may become invalid.	• Working With Libraries • Administrating Libraries
Working With Libraries Administrating Libraries	**note_library_version_deletion**	If you delete a library version in the administration that is still in use, the implementation of all affected services may become invalid.	• Removing Libraries • Administrating Libraries
Working With the Explorer	**create_project**	Enter a name for the service in the pop-up window **Create New Service**. Click **Save**.	• Folders • Services
Working With the Explorer	**folder_created**	The new folder is created and displayed in the tree.	• Namespaces • Folders • Process • Forms
Working With the Explorer	**import_limitations**	• You cannot import newer versions to a system having an older version installed. • There are no migrations if importing Designer content from versions prior to Designer 21.1.3 (2.0.165). This may result in the imported service not working. In the future, importing data from a version older than 21.1.3 will be disabled.	• Exporting and Importing Explorer Contents

Working With the Explorer	**note_access_restrictions_on_several_levels**	You can set access restrictions on several levels: • On a **namespace**: Namespace permission can be set in the User Management. These permissions have the highest priority. If a namespace with e.g. **read-only** permissions is assigned to the user in the User Management, You cannot override this permission in the Designer. For further information refer to the **Scheer PAS** Administration Guide. • On a **folder**: Folder permissions affect all sub-elements of the folder, but you can assign permissions to a service within the folder independently. • On a **service**: Service permissions only affect the corresponding service.	• Folders • Services
Working With the Explorer	**note_explorer_name_restrictions**	**Restrictions on Element Names** The only restriction for folder names is that they must not be empty. To all other elements in the Explorer, the following name restrictions apply: Element names... • ... must not be empty. • ... must not contain spaces. • ... must not start with numbers. • ... must not end with a period (**.**). • ... must not contain one of the following characters: **<**, **>**, **:**, **"**, **/**, ****, **|**, **?**, ***** • Furthermore, the following strings must not be used as element names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9.	• Working With the Explorer • Folders • Services
Working With the Explorer	**open_a_service**	To open a service, click on the service name in the explorer tree. 1. The Service tab will open. 2. The service details will be displayed on the right side of the tree. To access the service, click on the the explorer tab icon or on the service details area to close the explorer tab. Refer to Service Panel for further information.	• Working With the Explorer • Folders • Services
Working With the Explorer	**project_created**	The new service is created and displayed in the tree.	• Folders • Services
Working With the Explorer	**tip_profiles_admin_guide**	Refer to the Administration Guide for further information on the creation of namespaces.	• Working With the Explorer • Namespaces
Working With the Explorer	**version_considerations**	By exporting and importing, you can transfer Designer contents between PAS installations. These installations may have divergent versions. As of Designer 21.1.3, import will perform migrations on the imported service if the data to be imported is **not older than Designer 21.1.3** (2.0.165). If there is a version gap of more than one version, multiple migrations will be applied if necessary.	• Exporting and Importing Explorer Contents
Working With the Explorer	**whats_a_service**	A **Service** contains one executable micro service. Services are created in the Explorer tree. The necessary settings for your service such as the service properties, the BPMN model, the data model, used libraries and forms are managed within the service. You need to open a service to access its contents via the Service panel, refer to Service Panel for details.	• Working With the Explorer • Services

Working With the Explorer Modeling BPMN	**cockpit_tip**	For further information about the Cockpit, tiles and groups refer to the BPaaS Guide.	• Namespaces • Folders • Services • Libraries • Process • BPMN Model • Forms • Single Form
Working With the Explorer Modeling BPMN	**create_folder**	Enter a name for the folder in the pop-up window **Create New Folder**. Click **Save**.	• Namespaces • Folders • Process • Forms
Working With the Explorer Modeling BPMN	**paste_url**	You are now able to paste the URL, for example to an email.	• Namespaces • Folders • Services • Libraries • Process • BPMN Model • Forms • Single Form
Working With the Explorer Modeling BPMN	**toast_confirmation**	The action is confirmed by a toast message.	• Namespaces • Folders • Services • Libraries • Process • BPMN Model • Forms • Single Form