# PS Value Additions Library

## Description

The PS Value Additions Library is a collection of Teamcenter custom methods that may bring value to a customer implementation project. It currently contains a number of handlers, runtime properties and lower level utilities, such as a framework for logging. The plan is to add more and more useful methods to this library.

Hopefully, a library such as this can lower the implementation time at a customer. Since its code being re-used and maintained over time it will be more stable than customization done in a customer project.

Please contact me if you have done an implementation previously that may add value to this library.

## Contact

Please feel free to contact me if you have any questions or would like to make contributions to this library.

My contact details are:

Mattias Jonsson / [jonsson.mattias@siemens.com](mailto:jonsson.mattias@siemens.com)

## Installation

### Compatibility

So far, this library have only been built and tested on the Windows platform with Teamcenter 9.x and Teamcenter 10.x. There is no reason this library cannot be built targeting other platforms and Teamcenter versions with minor adjustments.

### Pre-requisities

Since different Teamcenter versions are used at customers, there is no binary pre-built distribution of this library. You will need the following components in order to build this library for your specific platform and Teamcenter distribution.

* This library code
* Microsoft Visual Studio C++ compiler (Make sure you have the correct VS version for your Teamcenter distribution)
* Teamcenter Corporate Server (Make sure the “include” folder is located in the Teamcenter root folder)

### Steps

* Copy and unzip the PS Value Additions Library zip-file to your build host.
* Edit the build script in a text editor
  + For Teamcenter 9.x, edit the “build\_tc9.bat” script.
  + For Teamcenter 10.x, edit the “build\_tc10.bat” script.
  + Edit the “MSDEV\_HOME” environment variable to point to the Visual Studio C++ compiler home folder. Hint, the folder path shall end with “…\VC”.
  + Edit the “TC\_ROOT” environment variable to point to the Teamcenter root installation folder.
  + Edit the “TC\_DATA” environment variable to point to the Teamcenter data folder.
* Open a command prompt and change directory to where the build script is located.
* Type “make” and then press enter.
* If not errors are reported during the build operation, type “make install” and then press enter. This will install the binaries into the “TC\_ROOT\bin” folder.
* Edit the file “TC\_ROOT\lang\textserver\en\_US\ue\_errors.xml”.
  + Add the line “<error id="100">%1$</error>”
* Log into Teamcenter as a dba user.
  + Add the value “libpsadds” to the preference “TC\_customization\_libraries”.
* Installation complete!!!

## Workflow Handlers

### Rule Handlers

#### PS2-check-in-process-RH

##### Description

This rule handler checks so that no targets of the current process are targets to another active process.

##### Syntax

PS2-check-in-process-RH

##### Arguments

No arguments

##### Placement

Place on the “Perform” action of the root task.

##### Restrictions

No restrictions

#### PS2-check-initiator-RH

##### Description

This rule handler checks so that the initiating user has the correct organization membership to run the current process.

##### Syntax

PS2-check-initiator-RH [–current\_user=<user1>[,<user2>…]] [–current\_group=<group1>[,<group2>…]] [–current\_role=<role1>[,<role2>…]] [–current\_full\_group=<full\_group1>[,<full\_group2>…]] [–current\_effective\_role=<role1>[,<role2>…]]

Note: At least one of the arguments have to be entered.

##### Arguments

**current\_user**

Comma separated list of user ids allowed to run current workflow.

**current\_group**

Comma separated list of groups allowed to run current workflow.

**Note**: The users currently logged in group have to match a value in the list.

**Note**: The group matched is the leaf-group in the organization. Please see “current\_full\_group” if you want to match on the whole organization group path.

**current\_role**

Comma separated list of roles allowed to run current workflow.

**Note**: Please see “current\_effective\_role” if you want to match against all the role memberships of the current user.

**current\_full\_group**

Comma separated list of groups allowed to run current workflow.

**Note**: Input has to be the full organization path, for example “level1.level2.level3”.

**current\_effective\_role**

Comma separated list of roles allowed to run current workflow. This method will check against all available role memberships of the current user.

##### Placement

Place on the “Perform” action of the root task.

##### Restrictions

No restrictions

#### PS2-check-privileges-RH

##### Description

This rule handler checks so that the initiating user has the correct privileges to run the current process.

##### Syntax

PS2-check-privileges-RH [–privileges=READ,WRITE,DELETE,CHANGE,PROMOTE,DEMOTE,COPY, IMPORT,EXPORT,TRANSFER\_IN,TRANSFER\_OUT] [–owning\_user] [–owning\_group] [–object\_types=<type1>[,<type2>…]] [–include\_statuses=Working[,<status1>,<status2>…]]

**Note**: At least one of arguments “privileges”, “owning\_user” or “owning\_group” is mandatory.

##### Arguments

**privileges**

Comma separated list of access identifiers describing what access the user have to have on selected targets to run the current process.

**owning\_user**

A flag dictating that the user has to be the owner of selected targets to run the current process.

**owning\_group**

A flag dictating that the user has to be in the owning group of selected targets to run the current process.

**object\_types**

Comma separated list of object types that is to be checked against.

**Note**: If argument is left out all object types attached as target to the process are being checked.

**include\_statuses**

Comma separated list of status types. Targets are only checked if they contain any of these status types in the “release\_status\_list” reference property.

**Note**: The “Working” string is reserved for describing that an object with no status are also to be checked.

**Note**: If argument is left out all objects attached as target to the process are being checked disregarding attached statuses.

##### Placement

Place on the “Perform” action of the root task.

##### Restrictions

No restrictions

#### PS2-check-child-structure-RH

##### Description

This rule handler checks so that the structure one level beneath a process target is valid for the current process.

##### Syntax

PS2-check-child-structure-RH –rev\_rule=<revision rule> –bomview\_type=<bomview type> [–include\_target\_types=<type1>[,type2…]] [–include\_child\_types=<type1>[,<type2>…]] [–allow\_if\_target]

**Note**: Arguments “rev\_rule” and “bomview\_type” are mandatory.

##### Arguments

**rev\_rule**

The name of a revision rule to be used when expanding the child structure of the process targets.

**bomview\_type**

The bomview structure type to expand.

**include\_target\_types**

Command separated list of item revision types attached as target to the process to include for validation.

**include\_child\_types**

Command separated list of item revision types in child structures to include for validation.

**allow\_if\_target**

If a child structure item revision is not valid, make it valid if it is attached as a target to the current process.

##### Placement

Place on the “Perform” action of the root task.

##### Restrictions

No restrictions

### Action Handlers

#### PS2-move-attachments-AH

##### Description

Action handler that moves target attachments to reference attachments or vice versa.

##### Syntax

PS2-move-attachments-AH –operation=target\_to\_reference|reference\_to\_target [–include\_types=<type1>[,<type2>…]

**Note**: Argument “operation” is mandatory.

##### Arguments

**operation**

Can be any of the value “target\_to\_reference” or “reference\_to\_target” depending on direction.

**include\_types**

The attached target object types to be included in the operation.

**Note**: If unset, all objects will be included in the operation.

##### Placement

No restrictions

##### Restrictions

No restrictions

#### PS2-timer-start-AH

##### Description

Action handler that starts a named timer watch instance.

##### Syntax

PS2-timer-start-AH –marker=<marker name>

**Note**: Argument “marker” is mandatory.

##### Arguments

**marker**

Name of named timer watch instance.

##### Placement

No restrictions

##### Restrictions

Site preference “PS2\_PerfTimer\_Enabled” must be set to true for markers to be created.

#### PS2-timer-stop-AH

##### Description

Action handler that stops a named timer watch instance.

##### Syntax

PS2-timer-stop-AH –marker=<marker name>

**Note**: Argument “marker” is mandatory.

##### Arguments

**marker**

Name of named timer watch instance.

##### Placement

No restrictions

##### Restrictions

Site preference “PS2\_PerfTimer\_Enabled” must be set to true for markers to be created.

#### PS2-create-dispatcher-request-AH

##### Description

Action handler that stops a named timer watch instance.

##### Syntax

PS2-create-dispatcher-request-AH –provider=<provider name> -service=<service name> [-primary\_type=<primary type>] -secondary\_type=<secondary type> [-priority=<priority>] [-arguments=<argument1>=<value2>[,<argument2>=<value2>…]] [-request\_type=<request type>]

**Note**: Argument “provider”, “service” and “secondary\_type” is mandatory.

##### Arguments

**provider**

Provider name of the called dispatcher module.

**service**

Service name of the called dispatcher module.

**primary\_type**

The dataset types to be “translated” by the called dispatcher module.

**Note**: If primary\_type is not set, a NULLTAG will be sent as argument for the primary type.

**secondary\_type**

The process target types to be processed by the called dispatcher module.

**priority**

The priority of the called dispatcher job. Must be an integer value.

**arguments**

Extra arguments sent to the called dispatcher module.

**request\_type**

Name of the request type.

**Note**: If not entered, default value of “ON\_DEMAND” will be supplied.

##### Placement

No restrictions

##### Restrictions

No restrictions

#### PS2-copy-task-properties-AH

##### Description

Action handler that copies the value of a number of properties on a process task to its targets objects.

##### Syntax

PS2-copy-task-properties-AH –task\_properties=<property1>[,property2…] –target\_properties=<property1>[,property2…] –include\_type=<include type> -attachment\_type=target|reference

**Note**: Argument “task\_properties”, “target\_properties”, “include\_type” and “attachment\_type” are mandatory.

##### Arguments

**task\_properties**

A list of properties on the current task that is to be copied to the target objects.

**target\_properties**

A list of properties on the target objects to be written to.

**include\_type**

The target object type to be written to.

**attachment\_type**

Process target type list to process. Valid values are “target” or “reference”.

##### Placement

No restrictions

##### Restrictions

Currently supports single/array property types “int”, “float”, “string”, “reference” and “untyped reference”.

## Other functions

### Runtime Properties

#### Naming rules

##### Description

Teamcenter only allows naming rules on a very restricted set of properties. This method will allow one to assign a naming rule on any property.

##### Syntax

The method is configured by the site preference “PS2\_Naming\_Rules”. It is an array preference of string type. Each line in the preference is a delimited string with a list of parameters. The format is:

<object\_type>:<property\_type>:<property\_name>:<regular\_expression>:<length>:<error\_message>

##### Arguments

**object\_type**

The object type which holds the property which is to be naming rule controlled.

**property\_type**

The property type of the property which is to be naming rule controlled.

**Note**: Supported property types are “int”, “float”, “double”, “string”.

**property\_name**

The name of the property which is to be naming rule controlled.

**regular\_expression**

The regular expression describing the naming rule pattern.

**Note**: For example, a naming rule that makes sure that the user enters a valid e-mail address could be “^[a-z0-9\_-]+@([a-z0-9-]+\.)\*[a-z0-9-]+\.[a-z]+$”.

**length**

Restricts the input length of the field.

**Note**: Useful when using an COTS attribute that needs to be restricted.

**error\_message**

Error message to be presented to the user if naming rule does not find a match.

#### Referencer Properties

##### Description

There is no COTS function in Teamcenter that one can use to display the backwards reference of a relation property in a property. With this function you can create a backwards reference on any reference or relation property.

##### Syntax

The method is configured by the site preference “PS2\_Referencer\_Properties”. It is an array preference of string type. Each line in the preference is a delimited string with a list of parameters. The format is:

<object\_type>:<property\_name>:<target\_relation>:<filter\_type1>[;<filter\_type2>…]

##### Arguments

**object\_type**

The object type which holds the runtime property on which to attach the reference value.

**property\_name**

The property name on which to attach the reference value.

**target\_relation**

The relation to follow in order to resolve the backwards reference.

**filter\_type**

A list of primary object types to show.

**Note**: If “filter\_type” is set to value “\*” all primary object types will be shown.

### Debugging

The debugging tools are in a standalone project that is statically linked in to the utility library. If one wish to re-use functionality inside this project at a customer project, it can easily be added to a solution and linked together with customer code.

#### Logging

##### Description

A configurable logging framework that lets you log different severity levels and also configure which targets to log to.

##### Syntax

The logger is configured by two site preferences. The preference “PS2\_Log\_Mask” defines the severity levels that are to be logged. The preference “PS2\_Log\_Streams” defines the targets to log to.

##### Arguments

**severity**

There are 4 different severities that can initiate a log entry. Each of them is represented by an integer number. The severity levels are:

* Error = 1
* Warning = 2
* Info = 4
* Debug = 8

For example, an integer value of 3 assigned to PS2\_Log\_Mask preference would enable logging events for severities Error and Warning.

**stream**

There are 3 different streams that a log entry can be written to. Each stream is represented by an integer number. The streams are:

* Standard Error = 1
* Standard Out = 2
* System Log = 4

For example, an integer value of 5 assigned to PS2\_Log\_Streams preference would enable logging to System Log and Standard Error.

#### Performance Timers

##### Description

The performance timers can be used to debug custom code methods to find time consuming methods. All utility methods supplied by this library have performance timer markers assigned. If this function is enabled, detailed timing statistics is presented when exiting the Teamcenter server.

##### Syntax

The performance timer is toggled on or off by a site preference which is called “PS2\_PerfTimer\_Enabled”.

##### Arguments

**enable**

Set the “PS2\_PerfTimer\_Enabled” site preference to “TRUE” in order to enable functionality.