

UI Commands in PSNext 3.0

02/15/09 – Revision 1.0

This document, as well as the software described in it, is furnished under license and may only be used or copied in accordance with the terms of such license. The information in this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Sciforma. Sciforma assumes no responsibility or liability for any errors or inaccuracies that may appear in this document. No part of it may be reproduced or transmitted, in any form or by any means without the prior written permission of Sciforma. Copyright 2009.

Content

A -Introduction.....	3
B -UI Commands.....	4
1 -Definition.....	4
2 -Create new commands.....	5
2.1 -External commands.....	5
2.2 -Internal commands (API based).....	6
2.2.a -Create a new internal command.....	6
2.2.b -Passing parameters.....	7
2.2.c -More on the Jar URL field.....	8
2.3 -Menus and toolbars.....	9
C -Extend built-in commands.....	10
1 -Defining the Custom Command	12
D -Conclusion.....	14

A -Introduction

Customizing and creating new fields, forms, graphical interfaces and data structures make PSNext an easy to use tool for any corporate environment.

These capabilities in PSNext 3.0 have increased considerably and UI Commands are part of it.

Customization is now extended to PSNext commands. Users can customize the behavior of some built-in commands to better adapt them to their own processes and utilizations.

Additionally, administrators are able to create new menu items and toolbar icons and associate them to a user defined program that could even interact with the logged-in user by using PSNext's API.

Java programming tools will will provide the remaining features for further customization such as dialogs, window controls or even enhance Web services to dialog with third-party applications.

B -UI Commands

1 - Definition

UI commands help to extend built-in features of PSNext to answer particular needs of a corporation. UI commands are managed by PSNext's administrator under System/UI Commands.

PSNext 3.0 enhances five ways to customize commands:

- Create a new command,
- Extend a built-in command,
- Run an external application,
- Run a PSNext API program,
- Extend a built-in PSNext command.

2 - Create new commands

New commands can be created for utilization on either of the PSNext components where tool and menu bars can be customized: Resources, Planner and Portfolio Control.

According to the kind of code or application a new command will run it can either be considered as an "External" command or an "Internal" command.

2.1 - EXTERNAL COMMANDS

External UI Commands allow administrators to create menu items and tool bar buttons and associate an external application to be run. The external application could be located on the client's machine or on some other shared location which the PSNext Client has access.

External UI Commands are created under the "External" tab of the System/UI Commands view.

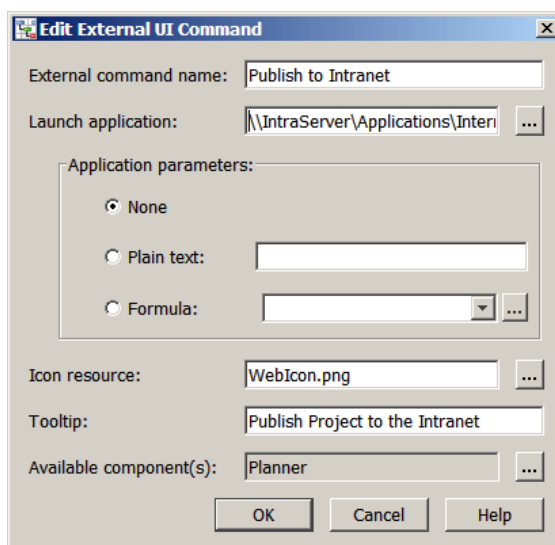


Fig 1. Settings of an External UI Command

EXTERNAL COMMANDS NAME	Name describing the command and label of the related menu item that will be created.
LAUNCH APPLICATION	File (.exe, .com) or scripts (.bat, .vbs) associated to run when the UI command is called.
APPLICATION PARAMETERS	PSNext can pass parameters to the external program. The parameters to be passed can either be “hard coded” or dynamically calculated by using formulas. The “current” context can be transferred this way to the launched application (the current Project, the current Resource, etc) <ul style="list-style-type: none"> ● None: No parameters should be passed. ● Plain text: Pass a fixed set of parameters. ● Formula: Formula whose result is passed as the parameters.
ICON RESOURCE	Fully qualified path to an icon resource (.gif, .jpg or .png). The icon will be used when displaying the new command in the toolbar. PSNext will use a default icon if this value is omitted
TOOLTIP	Text to display when mouse is over the related menu item or toolbar icon.
AVAILABLE COMPONENTS	Components where the UI Command menu and toolbar button will be available.

It is important to say that External UI Commands can only be run by PSNext clients under a Windows OS.

2.2 -INTERNAL COMMANDS (API BASED)

Instead of calling an external application, a new menu item or toolbar button can actually call a PSNext API program.

The first time a user runs a UI Internal command, the specified jar file will be downloaded from the server to the client machine and cached on the user's disk.

The PSNext API program will then be executed in the PSNext Client JVM. Subsequent use of the command will use the cached copy of the jar file on the client machine.

Since the API program knows how to manage PSNext's objects and methods, it will be able to interact with live data the user is working with. And since it is an independent program as well, it can provide with its own features, dialog boxes and additional coding to enhance the command.

A common utilization of a new internal command is the creation of new “working” projects from the Portfolio Control component.

2.2.α -CREATE A NEW INTERNAL COMMAND

Internal UI Commands are created in the “In process” tab under System/UI Commands.

Fig 2. New Commands

UI COMMAND NAME	Name describing the command. It will be used in the related menu item if no alias name is provided.
JAR URL	Location of the Jar file containing the PSNext API program .See “More on the Jar URL field” below
LISTENER CLASS NAME	Name of the class to call
NEW COMMAND	Indicates that a new command is to be created (menu item and toolbar button) and enables the controls to set the new command's properties.
ICON RESOURCE	Fully qualified path to an icon resource (.gif, .jpg or .png) contained in the Jar file. The icon will be used when displaying the new command in the Toolbar. PSNext will use a default icon if this value is omitted.
TOOLTIP	Text to display when mouse is over the related menu item or toolbar icon.
COMMAND ALIAS	Name of the related menu item. If omitted, the UI Command name will be used.
AVAILABLE COMPONENTS	Components where the UI Command menu and toolbar button will be available.

2.2.b -PASSING PARAMETERS

As for external commands, a set of parameters can be passed to the PSNext API program that will be run.

The lower section of the dialog box displays a spreadsheet presenting a list of fields which can be passed as parameters.

Currently, only the [Global] field can be passed as a parameter. This reference field contains a set of information on the context running the command.

For instance the [Global] field can access the [Current project] field which returns the currently selected project when the command is run.

API enhancements are provided in PSNext 3.0 to access the [Global] field. For more information on these enhancements, please read "Configuration/API in PSNext 3.0".

2.2.c -MORE ON THE JAR URL FIELD

During the development of the PSNext API program it may be difficult to constantly build and deploy the jar file. In this case the Jar URL field can be left blank as long as the PSNext API program classes are on the class path.

In order to avoid creating a PSNext security vulnerability, the Jar file must be digitally signed (all of PSNext's built-in client jar files are signed).

It is important that every user that will execute the command has access rights to reach the related jar file. This file could generally be hosted in a web or application server. Administrators can decide to host the jar file in PSNext's application server since they are certain that every PSNext user has access permission to it.

PSNext provides some substitution variables to easily find the URL of a jar file hosted by the PSNext's application server. They are as follows:

- &protocol
- &host
- &port
- &context

All of these substitution variables are case-sensitive and resolve to the URL components that a PSNext user used to start the client for the current PSNext session.

Using this approach, the following are some examples of URLs that an administrator could specify for the .jar file:

http://&host:8080/PSNextExtensions/ObjectiveProjects.jar

http://MyWebServer:&port/PSNextExtensions/ObjectiveProjects.jar

&protocol://&host:&port/PSNextExtensions/ObjectiveProjects.jar

However it is not recommended that the PSNext API program be part of the same deployment on the app server as PSNext itself. If this were done, the custom command .jar file would unfortunately vanish whenever the customer undeployed PSNext (e.g. when installing a maintenance release).

Note that jar files should be compiled each time a PSNext upgrade is undertaken.

2.3 - MENUS AND TOOLBARS

The Portfolio Control, the Planner and the Resources component have a menu item called “UI Commands” under the “Tools” menu.

This item displays a sub menu listing all the external and internal UI Commands that have been made available to the component (see the “Available Components” field in the UI command definition)

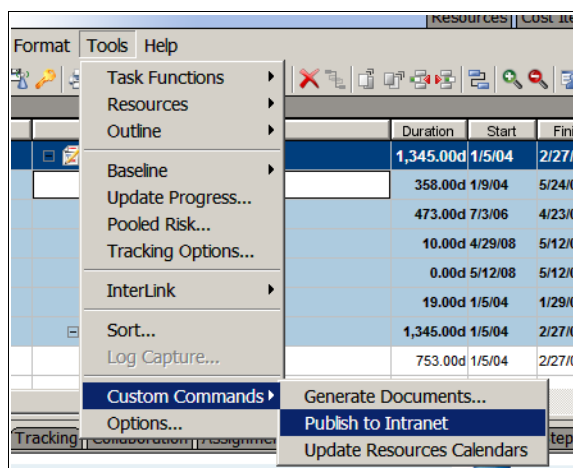


Fig 3. Available UI commands are listed in the Tools menu of the component

Additionally, since the toolbars of these components can be customized, the related icon of the new command can be displayed in either the Main bar, the Nav bar or the Info bar.

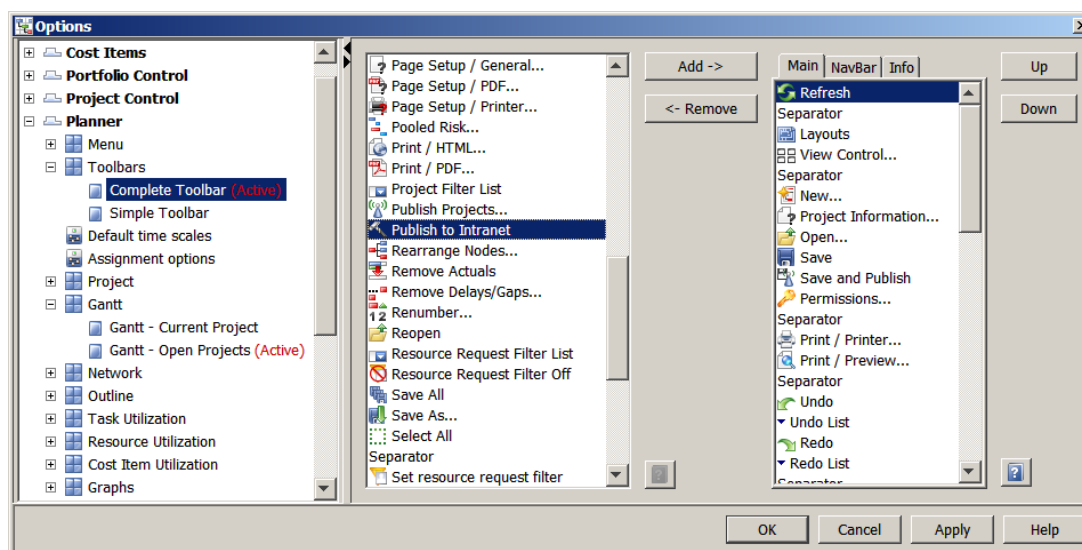


Fig 4. UI Command's icons can be displayed in the toolbar.

C -Extend built-in commands

The previous sections detailed how a brand new command can be created. A new menu item and tool bar button will enable users to run either an external program or a PSNext program.

In PSNext 3.0 some built in commands can be customized in many ways. An “In Process” UI Command definition can be related to a built-in command and will be run when the built-in command is called (by its built-in menu, toolbar icon or event).

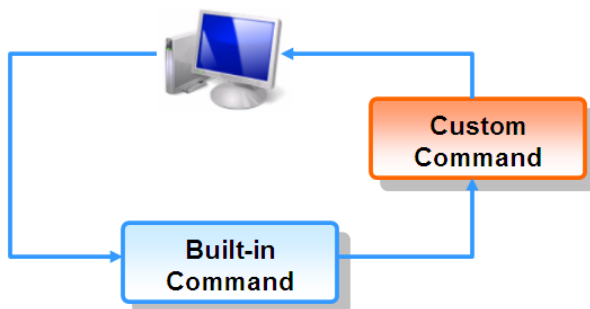
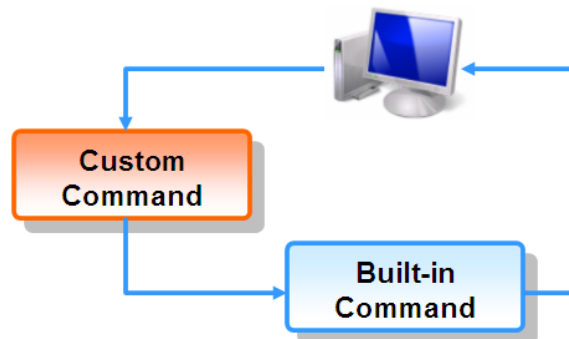
There are three possible ways to place the execution of a custom command against the execution of built-in command:

Before the built-in code is run

The custom command is executed first to pre-work the execution of the built-in command. Note that the built-in command might not be executed if the custom decides not to.

For instance publishing a project might require a given authorization from a third-party application. A custom command might be in charge to handle this authorization by using a Web service to dialog with the third party application.

If ever the authorization failed (project is not allowed to be published), the project publishing built-in code would not be called.



After the built-in code is run

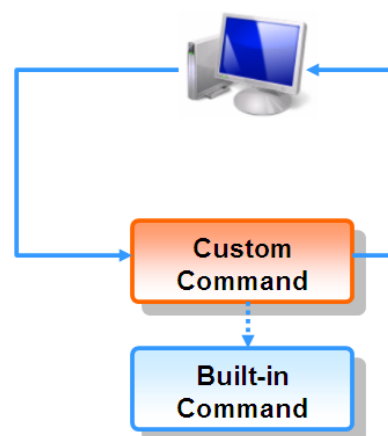
The custom command's execution can be placed right after the built-in command is successfully executed.

As an example, when a baseline is saved on a project, a custom command could extend this operation and save additional information into user defined fields for base lining purposes.

Instead of the built-in code

The previous cases are actual "extensions" of the built-in code. This last option is overloading the built-in operation since the custom command is executed instead of the built-in command. Even though the custom command can call the built-in command it's replacing, it might not to do it at all.

Lets suppose that a custom command would overload the project purge built-in command. Instead of actually purging the project, it will reset the project's permissions so that only Administrators get read permission on it.. The built-in purge command is never executed.



1 -Defining the Custom Command

To customize a built-in command, an “In-Process” UI Command definition is required. As previously seen this can be done in the “In-Process” tab under System/UI Commands.

The “Customized command” option will enable the controls to relate the command with a PSNext's built-in command (base command).

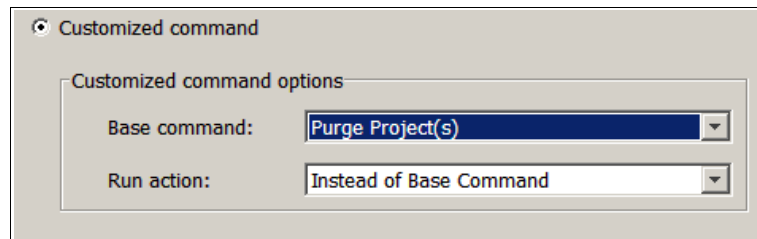


Fig 5. Built-in commands can be extended by an “In-Process” UI Command.

BASE COMMAND	Built-in command that will be customized. Table 1 presents a list of the built-in commands that can be customized in PSNext 3.0
RUN ACTION	Order of execution against the built-in command: Before, After or Instead the base command.

As previously seen, parameters can be passed to the the “In-process” commands by using fields.

Depending on the built-in command that is being extended the available parameters that can be passed will automatically be proposed in the input parameters spreadsheet.

For instance the built-in Project publish command displays a dialog box listing the projects to publish. If this command was to be extended, it would be useful to know the projects that the user selected for publishing in the built-in dialog box.

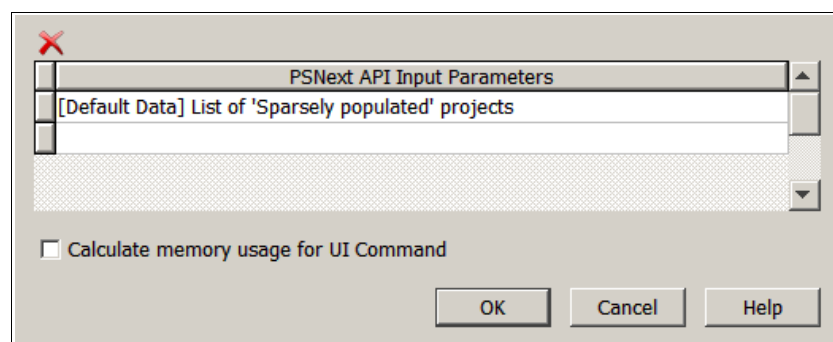


Fig 6. The available parameters to pass to the API program depend on the built-in command that will be extended.

Table 1 summarizes the built-in commands that can be extended

BUILT-IN COMMAND	COMPONENT	MENU ITEM	ICON	AVAILABLE PARAMETERS
Activate Project(s)	Planner Portfolio Control	File/Activate Projects...		[Global] [Default Data] List of 'Sparsely populated' projects
Close All Project(s)	Planner Portfolio Control Project Control	File/Close All		[Global] [Default Data] List of Projects
Close Project	Planner Portfolio Control Project Control	File/Close Project		[Global] [Default Data] Project
Deactivate Project(s)	Planner Portfolio Control	File/Deactivate Projects...		[Global] [Default Data] List of 'Sparsely populated' projects
New Project	Planner Portfolio Control Project Control	File/New... (Ctrl+N)		[Global] [Default Data] Project (run 'After' only)
Open Project(s)	Planner Portfolio Control Project Control	File/Open...		[Global] [Default Data] List of 'Sparsely populated' projects
Project Save As	Planner Portfolio Control Project Control	File/Save As...		[Global] [Default Data] Project
Publish project(s)	Planner	File/Publish Projects...(F9)		[Global] [Default Data] List of 'Sparsely populated' projects
Publish Resource(s)	Resources	Tools/Publish (F9)		[Global] List of Resource objects
Purge Project(s)	System	Edit/Delete (Del)		[Global] [Default Data] List of 'Sparsely populated' projects
Purge Resource(s)	System	Edit/Delete (Del)		[Global] [Default Data] List of Resource IDs
Remove Baseline(s)	Planner Portfolio Control	Tools/Baseline/Remove...		[Global] [Default Data] integer List [Default Data] Project [Default Data] Selected Task List
Save All Project(s)	Planner Portfolio Control Project Control	File/Save All		[Global] [Default Data] Project List
Save and Publish Project	Planner	File/Save and Publish		[Global] [Default Data] Project
Save Baseline(s)	Planner Portfolio Control	Tools/Baseline/Save...		[Global] [Default Data] integer List [Default Data] Project [Default Data] Selected Task List
Save Project	Planner Portfolio Control Project Control	File/Save (Ctrl+S)		[Global] [Default Data] Project

D - Conclusion

UI commands considerably increase the customization possibilities of PSNext environments. Totally new features can be developed by using the PSNext API and Java programming.

Additionally, some built-in PSNext's features can be customized to better fulfill corporate requirements or specific user needs.

Since these programs are run in the client side and use the PSNext API, total interaction with the user's live data is completely possible.