

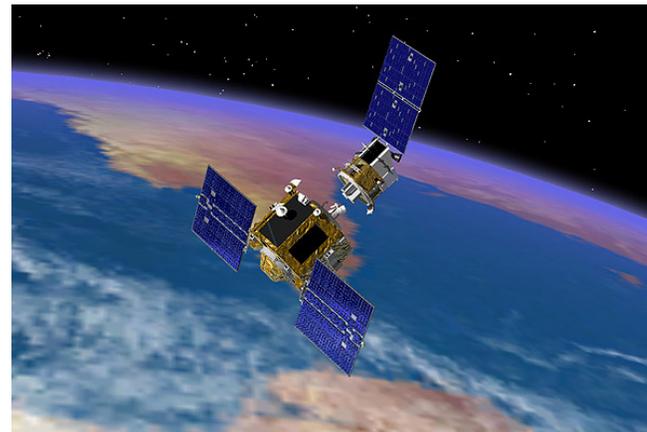


Information

Teamcenter Engineering Business Modeler Extension Rules

Mary Hoover
PDM Systems
June 3, 2008

- **Boeing is the world's leading aerospace company, with over 160,000 employees**
- **Two Business Units**
 - Boeing Commercial Airplanes
 - Integrated Defense Systems
- **Teamcenter Engineering is used in Huntington Beach, as well as other locations**



Overview

- **Business Modeler Extension Rules allow you to extend the behavior of Teamcenter with custom code or “canned” code**
- **Custom Extension Rules are added to custom libraries and assigned to operations on objects via the Business Modeler**
- **This presentation will show a simple example of creating and deploying a custom Extension Rule with Teamcenter Engineering 2005 SR1 / 2007**

Why Use Extension Rules ?

- **Once an Extension Rule has been deployed, it can be**
 - **added or removed from an operation/object**
 - **activated or inactivated**
 - **reordered**

at any time, *with no code changes.*

Features of Extension Rules

- **Extension Rules can be defined for Object Types, Properties, and User Exits**
- **Multiple Operations are available, depending on the object type**
- **Extension Points include Pre-Condition, Pre-Action and Post-Action**
- **An extension rule can be used for multiple extension points**
- **An extension rule can have a list of arguments, unique for each extension point**
- **Extension rules can be configured to execute in a specific order on an extension point**

Simple Example

- **Our simple example will be an extension rule that will be executed as a Post-Action on the Creation of an Item Revision**
- **There will be one argument at the extension point, *Name***
- **The value entered for the argument, *PLM World*, will be set as the *Item Revision Name***

Steps to Create a Custom Extension

- 1. Write the Code**
- 2. Create the Library**
- 3. Deploy the Library**
- 4. Define the Extension**
- 5. Assign the Extension**
- 6. Test the Extension**

Source Code

- **sample_extension_rules.h**
- **sample_extension_rules.c**
- **itk_macros.h**
- **compile_sample.bat**
- **link_sample.bat**



sample_extension_r
ules.zip

Write the Code

- **Create a function in ITK with two arguments**
 - `METHOD_message_t*` message
 - `va_list` args
- **Get the variable list of arguments**
 - `va_arg`
- **Get the parameters from the message**
 - `BMF_get_user_params`
- **Perform the action**
 - Your ITK code

Get the Parameters From the Message

- **Use `va_arg`**

```
pszItemID = va_arg( args, char *);  
pszItemName = va_arg( args, char *);  
pszItemType = va_arg( args, char *);  
pszRevID = va_arg( args, char *);  
ptagNewItem = va_arg( args, tag_t *);  
ptagNewRev = va_arg( args, tag_t *);  
tagMasterForm = va_arg( args, tag_t );  
tagRevMasterForm = va_arg( args, tag_t );
```

- **See the message header files to figure out which parameters will be provided through `va_arg` (*iman_msg.h*, *item_msg.h*, etc.)**

Message Parameters from Documentation

ITEM Messages - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites

Address Go Links

Main Page | Modules | Functions | Code Elements | Search | Deprecated

ITEM Messages

[Item]

Defines

- #define ITEM_create_msg "ITEM_create"
- #define ITEM_create_from_rev_msg "ITEM_create_from_rev"
- #define ITEM_create_rev_msg "ITEM_create_rev"
- #define ITEM_copy_rev_msg "ITEM_copy_rev"
- #define ITEM_deep_copy_msg "ITEM_deep_copy"
- #define ITEM_baseline_rev_msg "ITEM_baseline_rev"
- #define ITEM_copy_rev_to_existing_msg "ITEM_copy_rev_to_existing"

Define Documentation

```
#define ITEM_create_msg "ITEM_create"
```

Create a new item with initial working revision.

Parameters:

```
const char* item_id
const char* item_name
const char* type_name
const char* rev_id
tag_t* new_item
tag_t* new_rev
tag_t item_master_form
tag_t item_rev_master_form
```

Discussions Discussions not available for this document

Local intranet

Function to Get Arguments

- **Get the arguments using Sample Code in the Integration Toolkit Programmer's Guide**

```
int getArgByName(BMF_extension_arguments_t* p,
                int arg_cnt,
                const char* paramName)
{
    int i = 0;
    for (i=0; i < arg_cnt; i++)
    {
        if (strcmp(paramName, p[i].paramName) == 0)
        {
            return i;
        }
    }
    return -1;
}
```

Locking Objects

- **If there is any possibility of interaction of your extension and other custom code, be careful with locking**
- **If an input to your extension is an object that is already locked**
 - **Don't lock it again**
 - **Don't unlock it after you're done**

```
ITKCALL1( POM_ask_instance_lock( tagObject, &iLockToken ) );  
if( iLockToken == POM_modify_lock )  
...  
...
```

Sample Extension Rule Code

```
/* Get the parameters from the message. */  
tagItem = va_arg( vArgs, tag_t);  
pszRevID = va_arg( vArgs, char *);  
ptagNewRev = va_arg( vArgs, tag_t *);  
...
```

```
/* Extract the user arguments from the message */  
ITKCALL2( BMF_get_user_params(tMsg, &iParamCount,  
    &tInputArgs)
```

Sample Extension Rule Code (cont.)

```
/* Get the value of the Name argument. */  
if (iFail == ITK_ok && iParamCount )  
{  
    iIndex = getArgByName(tInputArgs, iParamCount,  
        NAME_ARG_NAME)  
    if (iIndex != -1)  
    {  
        strcpy(szNewName,  
            tInputArgs[iIndex].arg_val.str_value);  
    }  
    MEM_free(tInputArgs);  
}  
  
ITKCALL2( SAMPLE_SetNameOnRev( tagNewRev, szNewName ));
```

Create the Library

- **Compile your code using the compile script in the Samples directory**

```
%TC_ROOT%\sample\compile -debug -DIPLIB=none  
sample_extension_rules.c
```

- **Link your code using the link_custom_exits script in the Samples directory**

```
%TC_ROOT%\sample\link_custom_exits  
libSampleExtRules
```

Deploy the Library

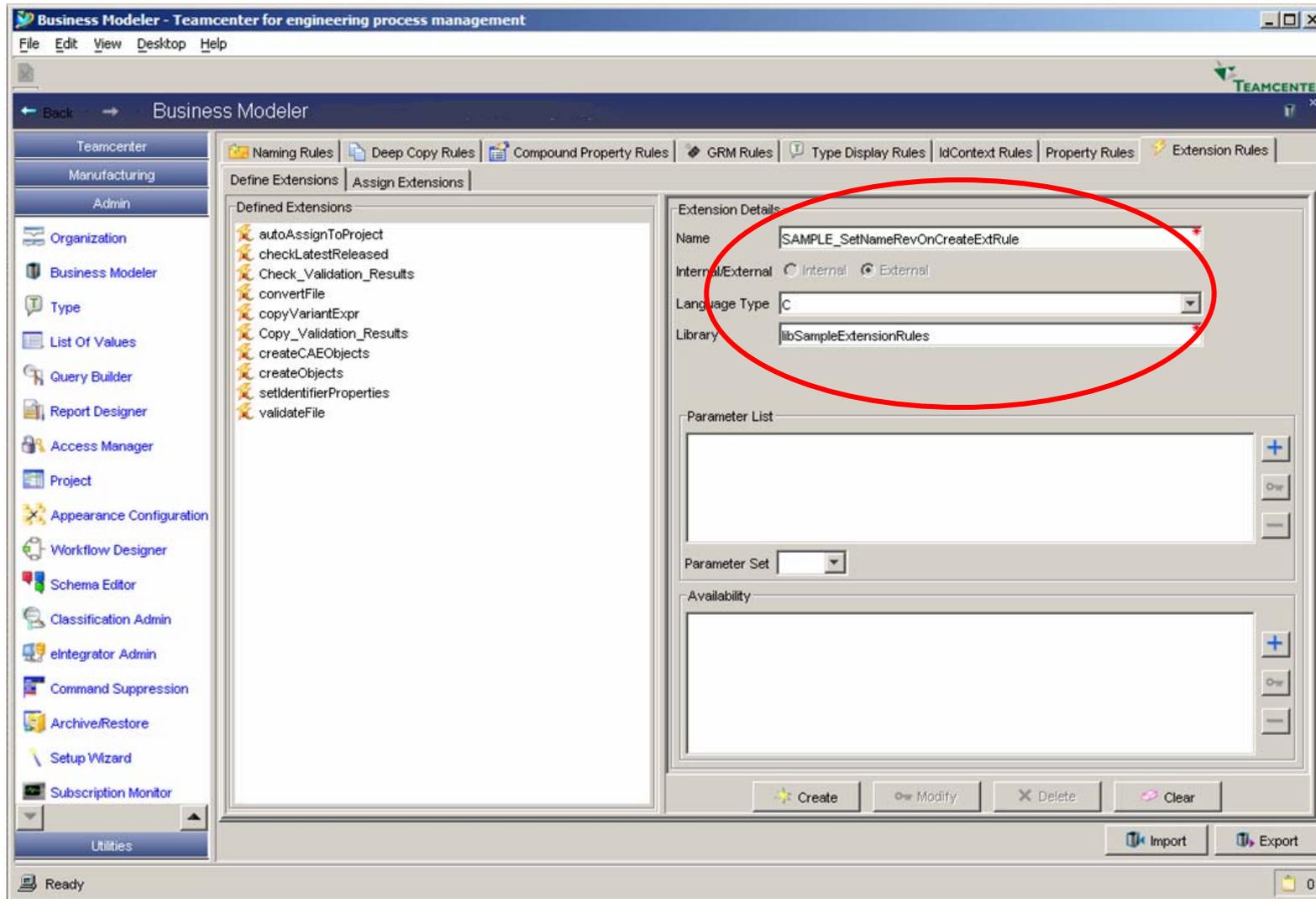
- **Move the Shareable library to the desired location**
- **Set the environment variable `BMF_CUSTOM_IMPLEMENTOR_PATH` to the path of your custom library**
- **Add multiple values for different platforms**

```
BMF_CUSTOM_IMPLEMENTOR_PATH=  
%IMAN_ROOT%\bin  
$IMAN_ROOT/lib
```

Define the Extension

- **Launch the Business Modeler Application**
- **Select the Define Extensions Tab**
- **Fill in the Extension Details**
- **Name and Language**
 - **C:** Name must match the function name
 - **C++:** Name must match `ClassName::MethodName` or `Namespace::ClassName::MethodName`
- **Library – Must match the name of your custom library**

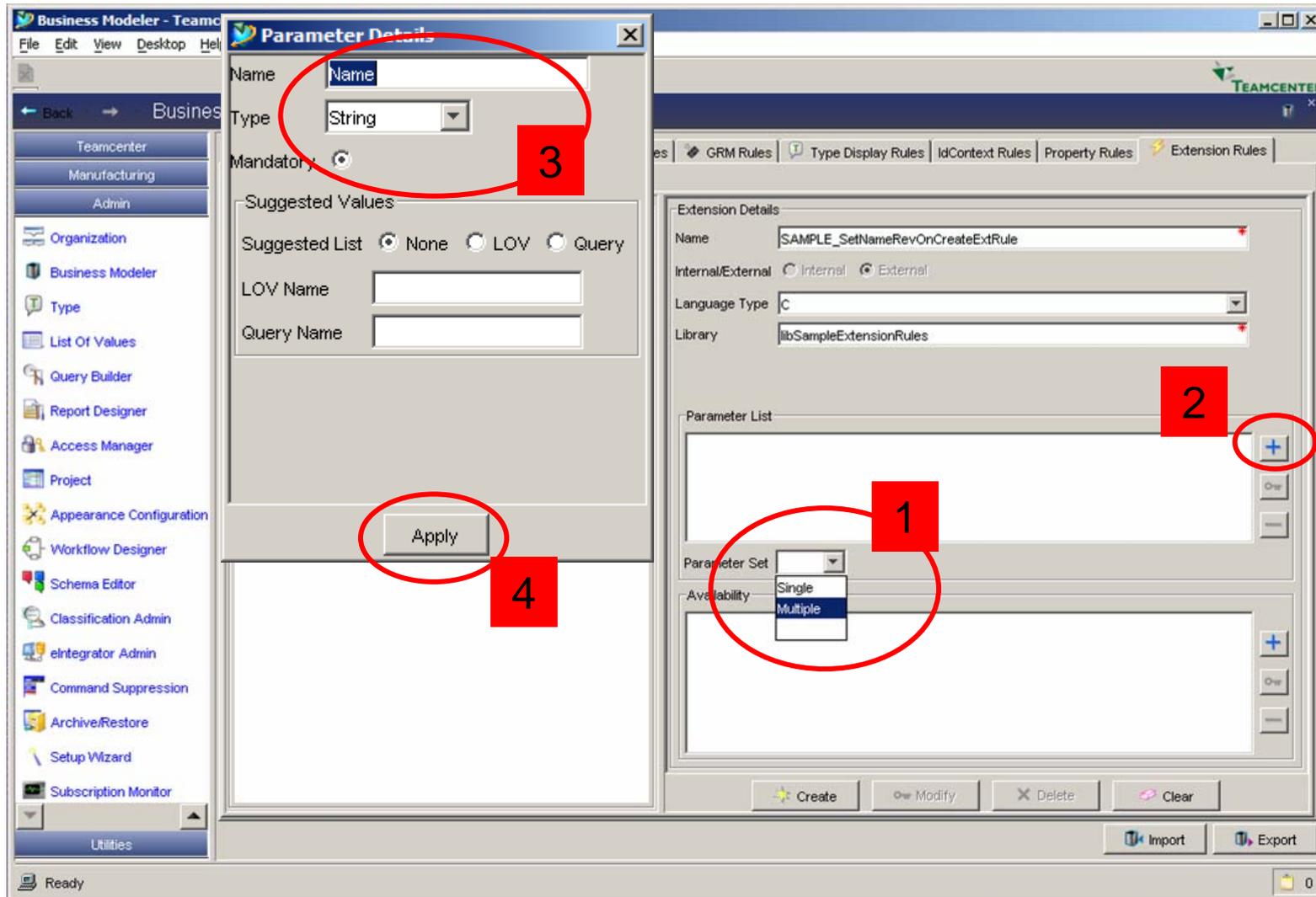
Define the Extension



Add the Parameter List

- **Select Single or Multiple Parameters (1)**
- **Select the + button to display the Parameter Dialog (2)**
- **Add the information for the Parameter (3)**
 - Name
 - Type (String, Integer, etc.)
 - Mandatory
 - LOV or Query for Suggested Values
- **Apply (4)**
- **Repeat for Steps 2-4 for each Parameter**

Add the Parameter List



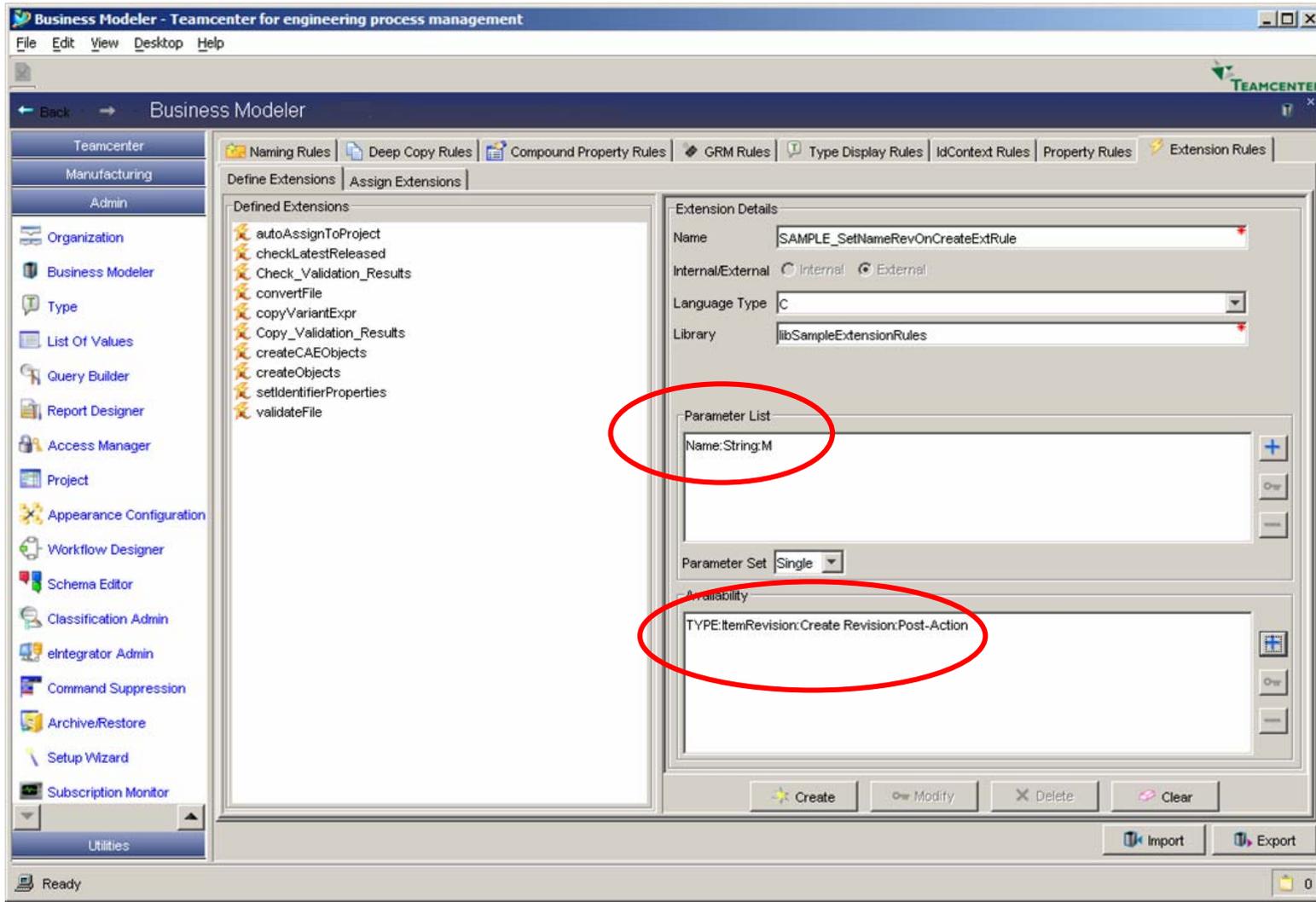
Add the Extension Point

- **Select + to add the Extension Point (1)**
- **Select the Object Type (2)**
- **Select Type or Property (3)**
 - **For Type, select Operation and Extension Point**
 - **For Property, select Property, Operation and Extension Point**
- **Create (4)**
- **Repeat Steps 3-4 for each Extension Point**

Add the Extension Point

The image shows two overlapping dialog boxes from the Teamcenter software. The left dialog, titled "Select Extension Point", has a tree view on the left with "ItemRevision" selected (circled in red with a "2"). Below the tree, "Type" is selected under "Select Type or Property" (circled in red with a "3"). The "Operation" is set to "Create Revision" and the "Extension Point" is "Post-Action" (circled in red with a "3"). The right dialog, titled "Extension Details", shows the "Name" field with "SAMPLE_SetNameRevOnCreateExtRule" (circled in red with a "4"). The "Internal/External" radio buttons are set to "External". The "Language Type" is "C" and the "Library" is "libSampleExtensionRules". The "Parameter List" contains "Name:String:M". The "Availability" section is empty (circled in red with a "1"). The "Create" button at the bottom is circled in red with a "4".

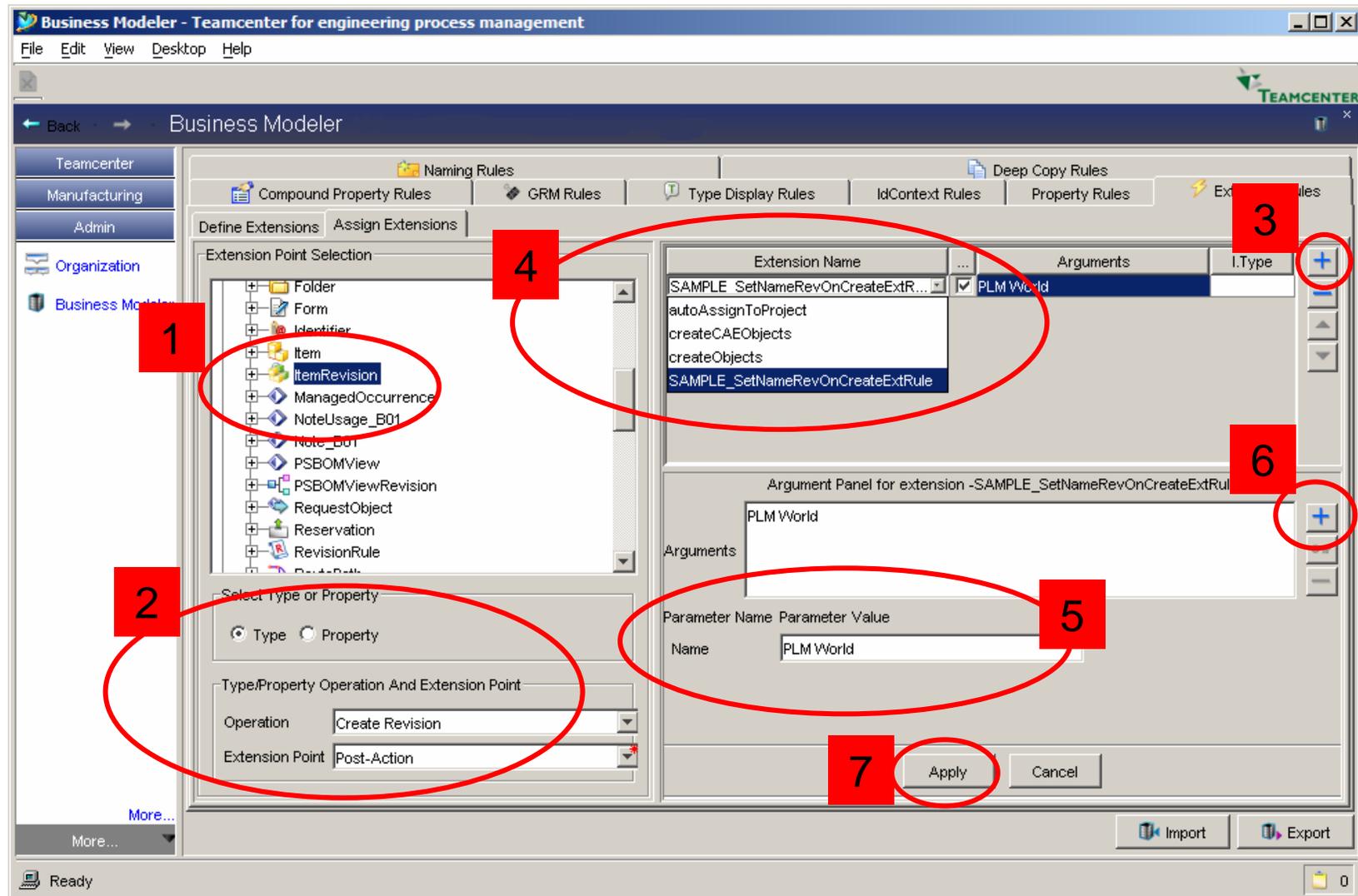
Defined Extension Completed



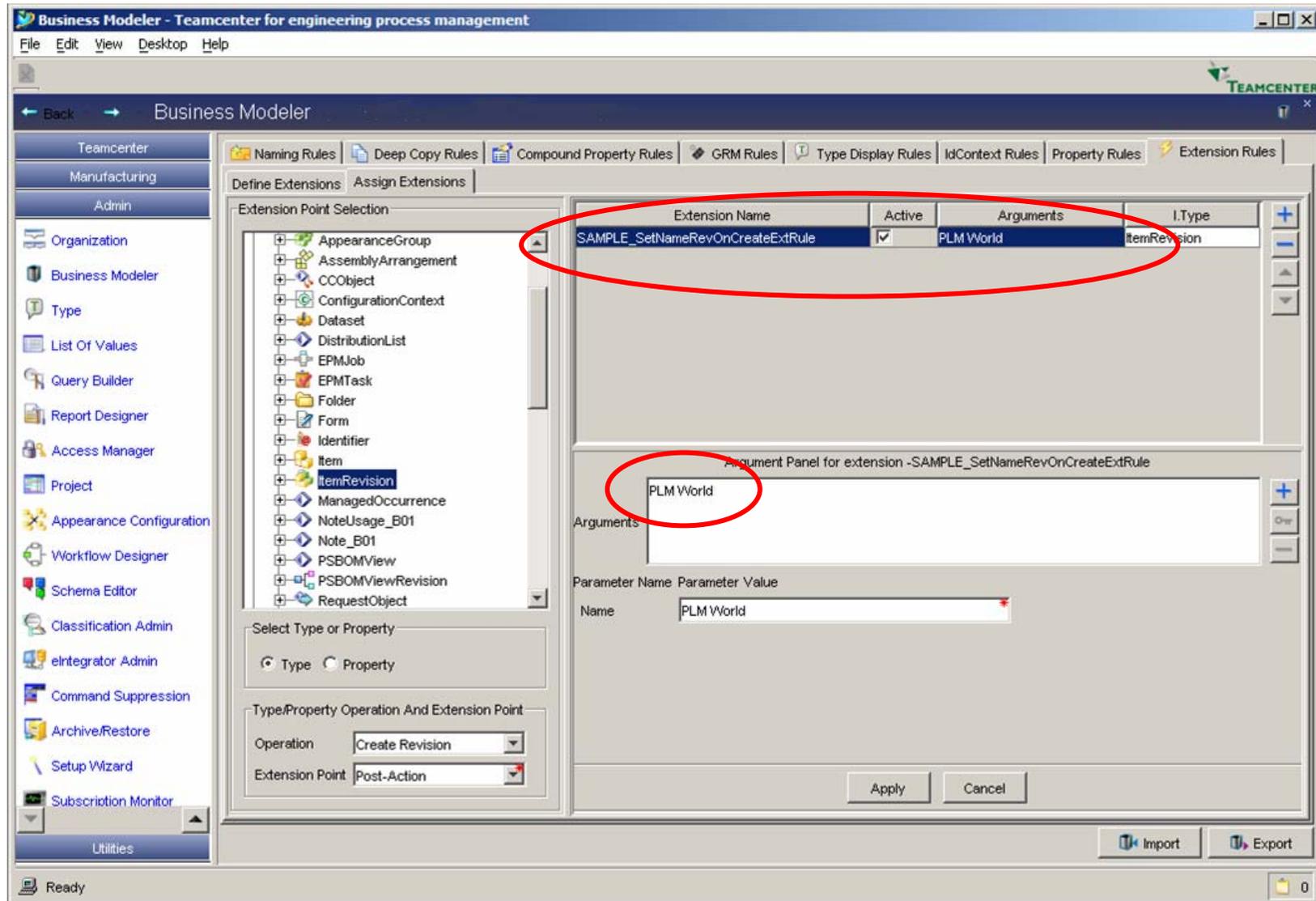
Assign the Extension

- **Select the Assign Extensions Tab**
- **Select the Object Type (1)**
- **Select Type or Property (2)**
 - **Select Operation and Extension Point for Type**
 - **Select Property, Operation and Extension Point for Property**
- **Select + to add an Extension (3)**
- **Select Extension name from list (4)**
- **Fill in each parameter (5) and select + (6)**
- **Apply (7)**

Assign the Extension



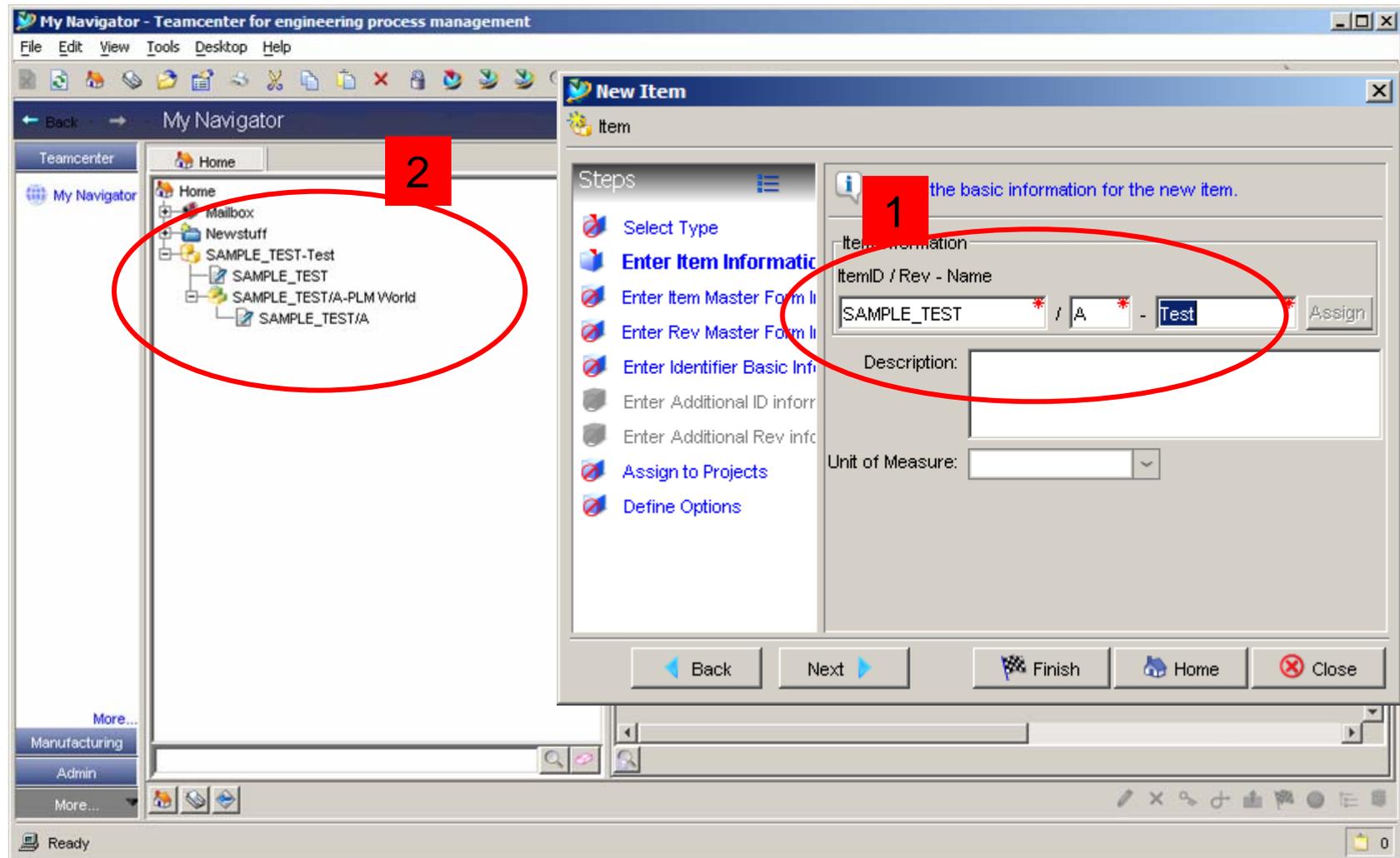
Completed Assigned Extension



Test the Extension

- **Create a New Item with a Name of “Test” (1)**
 - **The default Item Revision name will also be “Test”**
- **The Item Revision Name has been changed to “PLM World” by the Extension Rule (2)**

Test the Extension



- **If you want to extend the code on the creation of an Item Revision, remember to extend Create, Revise and Save-As**
- **Test with all User Interfaces, including Rich Client, Thin Client, CAD Client, etc.**
- **Remember to consider performance of your extension rule when performing bulk operations**
- **The definition and assignment of the Extension Rules can be exported via xml file and imported into another database**

- **General Business Modeler Documentation**
 - **Under Configuration**
 - Business Modeler Help
- **Business Modeler Programming Documentation**
 - **Under Customization**
 - **Integration Toolkit Programmer's Guide**
 - ITK Modules
 - System Administrator
 - Business Modeler
 - Adding New Extensions

Teamcenter Unified Documentation

Engineering, Operations & Technology | Information Technology

PDM Systems

- **General Business Modeler Documentation**
 - **Under Customizing Teamcenter**
 - Business Modeler IDE
- **Business Modeler Programming Documentation**
 - **Under Customizing Teamcenter**
 - **Integration Toolkit Programmer's Guide**
 - ITK Modules
 - System Administrator Modules
 - Business Modeler IDE
 - Adding New Extensions

