Volume

1

OPCAT SYSTEMS

OPCAT 4.0

User Guide

OPCAT 4.0 Server

OPCAT SYSTEMS

OPCAT 4.0 User Guide

Table of Contents

Chapter 1

Purpose of this Guide

The purpose of this guide is to allow experienced users of OPCAT 3 to work with the new features offered by OPCAT 4.0, which include transition to server architecture, the use of OPCAT Model Control (OMC), and cross-system dependencies design and tracking mechanism. Readers who are interested in learning how to model in OPCAT are referred to the "Modeling with OPCAT" tutorial which can be found on OPCAT website www.opcat.com.

What is new in OPCAT 4.0

OPCAT 4.0 is an Enterprise Edition, designed for use by enterprises. To this end, it includes the OPCAT Model Control (OMC) and messaging system modules, which allow multiple users to model multiple systems that interact with each other. OPCAT has been further enhanced to allow non-technical users to use the OPM system model to better understand and manage their teams and systems. This is done via Vision—OPCAT's Web-based reporting module. Additional features, including Exposing and Templates, were added to enable cross-system reuse modeling and model management.

OPCAT Model Control (OMC)

1. Concepts

OPCAT Model Control (OMC) is the information sharing module of OPCAT 4.0. At its core is a model repository, which stores OPCAT model files and possibly other types of files in the form of a file system tree—a typical hierarchy of files and directories. Any number of authorized clients can connect to the model repository, and then read from or, under certain condition, also write to these files. By reading, the client receives models which can be inspected. By writing, the client makes changes she or he made to a model available to other authorized clients

1.1 The OMC Organizational Models Repository

The models in OMC are stored in the organization's server as an organization-wide models repository. This is the single place for reliably maintaining files, which is managed by the organization's system administrator. OMC includes several directories, some of which are reserved for special models, such as templates.

1.2 The OMC Local Working Copy

The OMC **working copy** is an ordinary directory tree on your local machine, containing a collection of files, which you checked out from the repository. You can edit these files without any limitation. Your OMC working copy is your own private work area. OMC will never incorporate other people's changes, nor make your own changes available to others, until you explicitly tell it to do so.

After you have edited one or more model files in your working copy and verified that they work properly, OMC enables you to "commit" your changes, i.e., synchronize the edited files with the repository, by writing them back into the repository, so other authorized people working with you on the same project can use them too.

Note that not all the OPCAT model files stored on your local machine are part of your working copy environment. Only those files which you "checked-out" from the repository or which were explicitly saved in the "Working Copy" directories will be available for synchronization with the repository. We explain this in more detail in the sequel.

1.3 The Lock-Modify-Unlock Mechanism

It is all too easy for users to accidentally overwrite each other's changes in the repository. Therefore, all version control systems have to solve the same fundamental problem: how

will the system allow users to share information, but prevent them from accidentally stepping on each other's feet?

OMC uses a lock-modify-unlock mechanism to address the problem of many authors clobbering each other's work. Using this mechanism, the repository allows only one person at a time to change any given file. This exclusivity policy is managed via locks. When a file is checked-out by one user, no other user can edit this file. All the other authorized users can, however, view the checked-out file, edit it and save it under a different name.

1.4 Offline and Online Usage

OPCAT allows you to work online or offline. While working online, you will be able to work in coordination with the repository, check-out files, commit, etc. When you are disconnected from the repository, the files that were checked out to your computer will still be there, but you will not be able to commit them back into the repository until you reconnect. Note that unless you unlocked the files before disconnecting, all the files you checked out will remain locked for edit by other users. As explained in detail below, a file or a directory you created while being offline can be added to the repository when you are back online.

1.5 Credentials – User Name and Password

The repository is protected by a password that each user must have. You need to contact your system administrator in order to get your credentials—both user name and password. As long as you work with your local copies only, you will not be asked to provide your credentials. As soon as you try to access the repository, either by clicking OPCAT's Repository Browser tab or by performing any action involving the repository, a login screen will appear, requiring your credentials.

2. User Consoles

OMC provides three types of messages, which appear in the grid at the bottom part of the application: File Console, Admin Console, and Message Console. If you do not see those messages, you can go to View on the top bar and select the console you want to see. Conversely, if you do not wish to see those messages, you can go to View on the top bar and unselect the console you do not want to see. Following is a description of the content of each console.

2.1 File Console

The File Console, accessible by the File Console tab in the grid bottom part, enables you to see messages related to files or directories. Here you can find the message number, the action you tried to perform with the file or directory, the name of the file or directory on which the action was attempted, a message describing the results of this action, the author of the file or directory, and the revision number.

2.2 Admin Console

The Admin Console, accessible by the Admin Console tab in the grid bottom part, shows messages related to global information, which is not specific to a certain file or directory. The Admin Console also presents information about errors that occur while trying to perform actions at the models tabs or actions that involve the repository. Always check this console in case an action was not performed as anticipated.

2.3 Messages Console

The Message Console, accessible by the Message Console tab in the grid bottom part, allows authorized users to view all the activities performed on the server. It is a powerful tool for managers, who can use it to estimate the work done by their team members. The information includes message ID, the message itself, its sender (the initiator of this action), date, type, sub-type and severity (which for now is set as "information" for all messages). Here you will also find messages related to changes to Exposed things as will be explained in details later on.

3. Basic Controller

The left pane in OPCAT 4.0 provides access to the controls of OMC via two tabs (see Figure 1): the **Models** tabs, which include all the locally saved Working Copy, and the **Repository Browser** tab, which allows you to work with models stored at the Repository. Each tab includes several directories. Clicking on the key icon at the left-hand side of the directory opens its internal directories or files. Actions on a file or a directory can be achieved by right-clicking the file or directory's name. Information is frequently presented at tabs opened at the Bottom Grid.

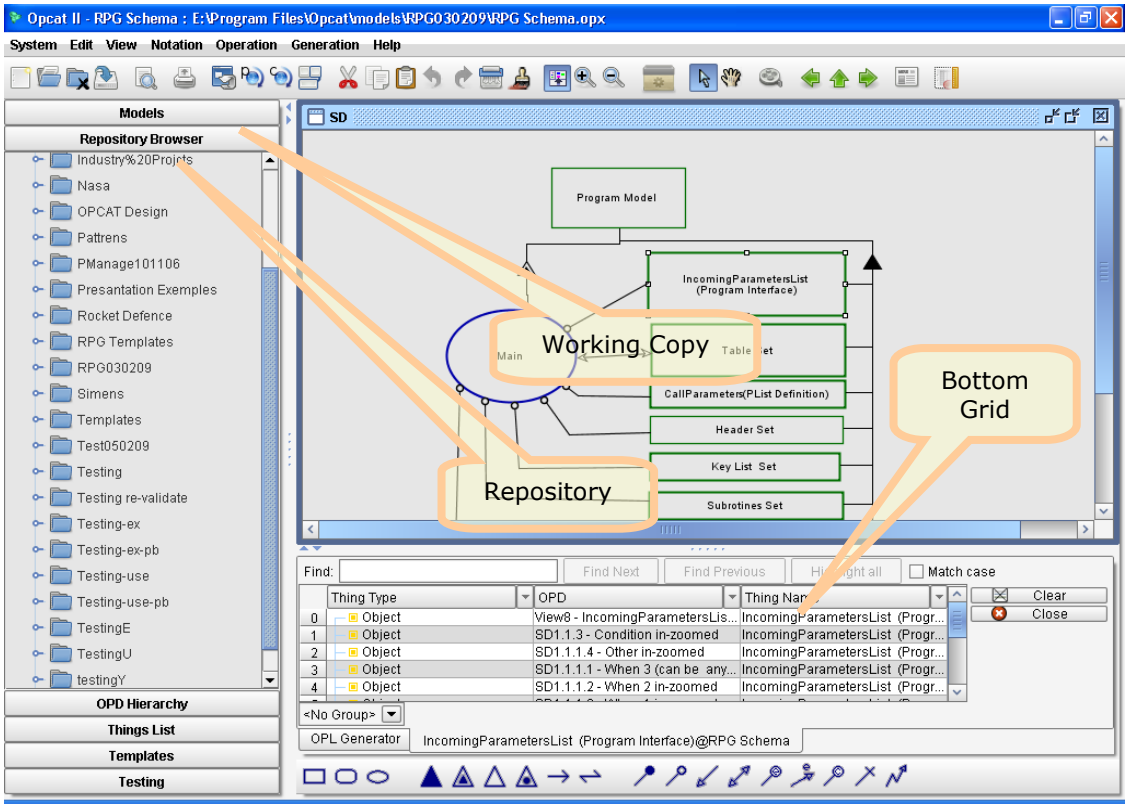

Figure 1. The Working Copy , Repository and Bottom Grid

4. Repository Browser Controllers

4.1 Checking Out

Checking out a directory amounts to copying the directory with all its files from the Repository to your local folder. A checked-out file will appear as LOCKED for other users and marked with the Locked-by-Me symbol when viewing it yourself in the Repository Browser.

To check out a directory, go to the **Repository Browser** tab, right-click on the folder you wish to edit, and select **Checkout**. Now go to the **Models** tab, where you will find the checked out directory (refreshing by right clicking on the root directory may be required).

To open a checked-out model, go to the Models tab and double click on the file, or right-click the model and select either **Open in Current Window** or **Open in New Window**.

Note that you can check out whole directories only. If the directory includes several files, all the files will be checked out. As explained below, files that were locked by other users will be checked out as read only. Remember that by checking out you lock all the files in the directory for editing by other users. Therefore, you should first unlock all the files you do not need to edit at this stage. In order to avoid confusion, you may also want to delete the unlocked files from you local copy. To unlock a file, select **Unlock** from the **Models** tab.

Caution: if you checked out a sub-directory by checking out its entire parent directory, do not check it out again by selecting the sub-directory. If you do so, you will create two copies of the sub-directory in you working copy environment.

Action Summary

Repository Browser → Right-click directory → Checkout → Models tab → [right-click root directory → refresh] → Double click model or Right-click model to open in a new window

4.2 Checking Out Locked Files

A small lock icon will appear next to each OPCAT model file that had been checked out by another user. You can find out more details about the identity of the user who checked out each file by right-clicking the file and selecting **Show Properties**. You can still check out the file. OPCAT will not allow you to save the files, but the **Save As** function will still be available.

4.3 Import

Import allows you to add files or directories which have not been added to the repository yet from your local machine directly to the selected directory in the Repository. To do this, in the **Repository Browser** right-click on the directory (or the root) to which you want to add the files, and select **Import**. At the selection pop-up, choose the directory you want to add from your local

machine. In order to add all the files in a directory simply select the directory. If you want to add the directory structure itself, a directory must be inside another directory which will be imported (second level import). For example, suppose directory D1 has a sub-directory called D2, in which there are two files. If you choose **Import** with D1, you will get D2 as part of the structure of the directory in your Repository. However, if you choose **Import** with D2, you will get the two files in the directory to which you imported in the Repository, but not the structure of D2.

ACTION SUMMARY

Repository Browser → Right-click directory → Import → Chose file or folder from the local machine

4.4 Delete

Delete allows you to delete empty directories and files from the repository. As expected, you cannot delete files that are locked by other users. In order to delete a file or a directory, go to the Repository Browser and right-click on the file of directory you want to delete and then select **Delete**. The file or directory that had already been checked out to your local working copy will remain on your local machine and will not be deleted automatically. As explained below, you can manually delete the leftover files from your local copy by clicking **Delete Locally** or **Update** (see the **Model Tab Controllers** section for more details). You cannot commit a deleted file or directory, even if they are still found in your **Models** tab. If you want to add deleted files back, you need to select **Update** by right-clicking on the file in the **Models** tab and then follow the steps described below for adding a file to the repository. Directories which were deleted cannot be restored. You will have to create a new directory and save the files in that directory.

ACTION SUMMARY

Repository Browser → Right-click empty directory or file → Delete

4.5 Revisions

Not Implemented Yet

4.6 Show Properties

This selection allows you to see the properties for a file or a complete directory. **Properties** include such information as lock statues, locked by, etc. The properties are shown at the bottom pane with a new tab, called **Repository Properties**. To see a file or directory's properties, right-click on the file or directory and select **Show Properties**.

ACTION SUMMARY

Repository Browser → Right-click directory or file → Show Properties

4.7 Refresh

Refresh updates the information presented on the screen. If you do not see changes you have done, try **Refresh** before proceeding with any further steps. To refresh, right-click on the root directory and choose **Refresh**.

ACTION SUMMARY

Repository Browser → Right-click on the root directory → Refresh

4.8 Add Directory

You may create new directory in your repository you can select Add Directory. To do this, in the **Repository Browser** right-click on the directory (or the root) to which you want to add the new directory, and select **Add Directory**. At the pop-up, select a name for the directory.

ACTION SUMMARY

Repository Browser → Right-click on a directory or root directory → Add Directory

4.9 Model Symbols in the Repository Browser

Figure 2 shows the symbols next to files in the **Repository Browser**.

ICON KEY	
	File is in normal position
	File is locked
	File is locked by me

Figure 2. Symbols in the Repository Browser

5. Model Controllers

5.1 Models Root Directories

At the top of the **Models** tab you will find two root folders, called **Models** and **Working Copy**. The folder **Models** is the root of your local machine, while **Working Copy** is the root reserved for your working copy directories.

5.2 Opening a Checked-out Model

To open a file that was checked out, simply double click on the file. This will open the file with the current running instance of OPCAT. OPCAT cannot open more than one model in one run instance, so if you wish to open several models, you can right-click the file and select **Open in New Window**, in which case a new run instance of OPCAT will be opened.

ACTION SUMMARY

Models tab → Double click file

Models tab → Right-click file → Open in Current Window

Models tab → Right-click file → Open in New Window

5.3 Adding a Directory from the Repository to "Working Copy "

You cannot add directories directly under your **Working Copy** root. As explained above, you can check out directories, which will appear under **Working Copy** or even add sub-directories to those directories.

5.4 Adding New Directories from Your Local Drive to The Repository

To add a new directory to your local **Working Copy** , which can later be added to the repository, go to the **Models** tab, right-click on the parent folder to which you want to add the new directory and select **Make Directory**. Recall that you cannot add it to the root **Working Copy** directory. A new directory will be created with a hazard icon. If you want to add this directory to the repository, right-click on the new directory and select **Add**. This will add the directory to the repository the next time you commit. Now, right click on the directory and select **Commit**. The directory will be added to the repository.

ACTION SUMMARY

Models tab → right-click a directory → Make Directory → Add → Commit

5.5 Adding Files From Your Local Drive To The Repository

To add a new file that was created locally or a file from the repository that was saved with another name (using **Save As**), the file must be saved in a working copy directory under **Working Copy** which already exists in the repository or is going to be added to the repository. After saving the file to an existing or a newly-created directory, using OPCAT's **Save** or **Save As** command, a small hazard icon will appear next to the file at the **Models** tab. Right-click on the file and select **Add**. A plus icon will appear next to the file, denoting that the file is scheduled to be added to the repository the next time you click **Commit**. If you right-click on the file again (or on the directory in which it is located) and click **Commit**, the file will be added to the repository. Remember that by committing the file you are not locking it. If you want the file to be locked, either select **Lock** or check it out from the repository after committing it.

ACTION SUMMARY

Save/Save As file → Go to Models Tab → right click on the file and select Add → Commit the file or directory

5.6 Commit

Once you are done changing your files, or at any time during your work, you can write those files back to the repository by right-clicking a file or directory icon and selecting **Commit**. **Commit** is done for the entire directory or the single file you selected. When committing, a popup window will suggest adding a commit note. In addition, you can decide whether to leave the file locked or to unlock it by marking **Unlock after Commit** at the same window. You can continue working on a committed file and then commit the changes again and again. Note that if you unlock the file and someone else has checked it out, you will not be able to commit the file again.

ACTION SUMMARY

Models tab → right click on file/directory

select Commit → add Commit Note → mark/unmark the "Unlock after Commit" checkbox

5.7 Lock and Unlock

As noted, when a user is working on a file, it is locked for editing by other users. These users can check out the file as read only and are not able to commit the file back to the repository. You can override this default and unlock a file which you locked yourself by right-clicking on the file and selecting **Unlock**. Note that by doing this, other users may now check out the file and lock it. If this happens, you will not be able to commit the file again to the repository. You can lock a file you unlocked by selecting right-click **Lock**. You will not be able to lock the file if someone else has already checked out the file and locked it while it was unlocked.

ACTION SUMMARY

Models tab → right click a file → Unlock

Models tab → right click a file → Lock

5.8 Delete File

Use **Delete File** when you want to delete a file from the repository. The file will be scheduled for deletion the next time you commit. A red symbol shall appear next to the file. Once you commit, the file will be deleted from the repository. After you commit the deletion, the file will appear in your **Working Copy**with a hazard icon, like a file that has not yet been added to the repository. You can delete the file completely by selecting **Delete Locally**. If you regret the deletion, you can add it back to the repository by clicking **Add** and **Commit**.

ACTION SUMMARY

Models tab → right click a file → Delete File → Commit → right click the file again → Delete Locally

5.9 Update

Update brings missing files which exists in the repository but are not in your **Working Copy** directory. **Update** works just like **Checkout** for the missing files. **Update** for a single file is visible but will be active only in the next version of **OPCAT Server**.

ACTION SUMMARY

Models tab → right click file or folder → select Update → right click on Models and refresh

5.10 Revert

If you decide that you want to throw away your changes and restart editing the file from the last version saved in the repository, right-click the file icon and select **Revert**. Remember

that by reverting, your local changes will be lost. The model brought from the repository will be automatically opened in OPCAT.

ACTION SUMMARY

Models tab → right click file or folder → select Revert → right click on Models and refresh

5.11 Cleanup

If OMC operation is interrupted (e.g., if the process is killed or the machine crashes), the log files remain on disk. By re-executing the log files, OMC can complete the previously started operation, and your working copy can get itself back into a consistent state. This is exactly what **Cleanup** does: it searches your working copy and runs any leftover logs, removing working copy locks in the process. To operate cleanup, select a file or directory at your models tab, press right-click and select **Cleanup**.

ACTION SUMMARY

Models tab → right click file or folder → select Cleanup → right click on Models and refresh

5.12 Add, Make Directory

See *Adding new Directories or new Files from your local drive to the Repository* above.

5.13 Delete Locally

You can use **Delete Locally** when there is a file on you local working copy which you do not need any more. This may be the case if you committed a file or directory, unlocked it and you are not planning to continue working on it, but it is still part of your local copy. In such a case, it is recommended that you delete those files or directories from your working copy. If you need the file or directory again, you can check out the latest version from the repository.

ACTION SUMMARY

Models tab → right click file or directory → select Delete Locally

5.14 Revision

Not implemented yet

5.15 Read Only Models

If you checked out models which were locked by other users, you will get those for read-only use only. The models will be marked with small lock icon in your Models tab. If the models were then

unlocked by the other user, you may now check them out for editing purposes. Before doing so, **Delete Locally** the files from your Working Copy environment.

ACTION SUMMARY

Models tab → right click file or directory → select Delete Locally→ open Repository Browser→ select a directory→ select Check-Out

5.16 Add Local Directory to "Models Tab"

You can add directories to your "Local Models" root if you want to use it to browse and open models on your local machine, but remember those directories will not be added to the repository. Only directories under "Working Copy" are under the OMC management and can be added to the repository. To add a local directory to Models right click on it and select Add Directory to Local List.

ACTION SUMMARY

Models tab → Right-click on Models → Add Directory to Local List → select the directory at the window

5.17 File Names

You may provide names to your models as you find applicable according to your organization conventions. Names can be in any language but should not contain the @ symbol.

Note that any model has a file name and System Name. The file name appears on the top. The System Name appears at the top of the OPD Hierarchy tab. The File Name and System Name may be or may be not identical. The reference in OPCAT is always to the System Name.

5.18 File Icons

Figure 3 shows the symbols next to files in the **Models** tab.

FILE ICON KEY	
	The file is in its normal state.
	The file has been changed locally (appears after local save), but was not committed yet.
	The file is scheduled to be deleted upon next commit.
	The file is new, it exists only on your local machine, and it is not scheduled to be added to the repository.
	The file is new, it exists only on your local machine, and it is scheduled to be added to the repository the next time you commit the file.

Figure 3. Symbols in the **Models** tab

Reuse and Dependency Tracking

Writing the same things over and over and then maintaining different programs doing the same thing is expensive. In order to save time and cost, you will be required to use existing code in most programs or systems you create. On the other hand, reusing the same things over and over creates a web of cross interdependencies between different parts of systems and code. Reuse must therefore be done wisely. If you don't know which other programs exist for you to reuse, it is less likely that you will choose the right program. Conversely, if you reuse other program and the developer in charge that program is not aware of this reuse and therefore changes the program, this can lead to dire unintended consequences.

OPCAT 4.0 was designed to help the designer reuse and unify code and systems while reducing potential reuse pitfalls. OPCAT 4.0 provides the designer with a list of all the available items, including programs, models, tables, and routines, along with a mechanism to coordinate between the **User**—the person who reuses an item, and the **Exposer**—the person who exposes that item, called the **Exposed Thing**.

1. Basic Definitions

Experienced OPCAT users know that when you copy a thing (object or process) from one OPD and paste it in the same OPD or in another one, you merely represent another appearance of the very same thing in the place where it was pasted. The copied thing symbol—ellipse or rectangle—can be considered as a pointer to the original thing. If the thing is a process, all its appearances will become activated when one of them does.

In order to reuse things, OPCAT 4.0 expands the support for successors of a thing, so you can create successors and not just appearances of the same parent thing.

We proceed with basic definitions to clarify the differences between parent, successor, appearance, and occurrence.

1.1 Thing Parent and Successor

An **object parent** is an abstract collection of objects – things that exist potentially or de-facto – which defines for that collection the set of attributes and their permissible, legal states and/or values.

When **Exposing** an object you are turning this object and its set of attributes into an object parent.

An **object successor** is a uniquely identifiable entity derived from an **object parent**, and therefore has the same set of features (attributes and operations) as the object parent from which it was derived. At any given point in time, each attribute of an object successor is in a defined permissible, legal state or value, or is in transition between such states or values.

When **Reusing** an object you are creating an object successor derived from the object parent, i.e the exposed object.

Different existing object successors can normally have different names, values and/or states at the same point in time, and each object successor can change its values and/or states or the values and/or states of any one of its attributes independently of any other successor of the same object parent.

A **process parent** is an abstract collection of processes which defines for that collection the set of attributes and their permissible, legal states and/or values.

A **process successor** is a uniquely identifiable entity derived from a **process parent**, which occurs at a specific point in time during the system's execution and transforms a specific set of one or more object successors.

Different process successors of the same process parent can exist in an OPM model, and each can be active at a different point in time, asynchronously, synchronously, sequentially or in parallel, transforming different sets of object successors.

1.2 Appearance

An OPM model is often complex and therefore. Except for trivial cases, is spread over a possibly large number of OPDs at various levels of detail. In order for a particular OPD to have some level of self-containment, some thing parent or thing successor needs to appear in that OPD once or more even though it is already depicted in one or more other OPDs. Therefore, the same thing parent or thing successor can appear in more than one OPD, giving rise to the concept of appearance, defined below.

Appearance is one of possibly many identical copies of the same **element** that may appear any number of times in various OPDs in the same OPM model, where **element** can be an **object parent**, a **process parent**, an **object successor**, or a **process successor**.

Any transformation that the object successor undergoes—its creation, destruction, or change of state—is reflected in all its appearances. Likewise, any occurrence of a process successor takes place simultaneously for all of its appearances.

You can create appearances in OPCAT by copying and pasting a thing.

1.3 Private and Public

If the parent thing from which you create successors is located in your model it is called a **Private** thing. If the parent thing is defined in another model it is a **Public** thing. The next sections explain how to use **Private** and **Public** things.

2. The Exposed Things List

All the things in your organization which are exposed are presented in the **Exposed Things List**. Examining this list from within a particular OPCAT model. you will also find there things which were **Privately Exposed** and therefore are visible only in this model. To view the **Exposed Things List**, click on the **Exposed** icon, which looks like this:  and is located at the top toolbar. In response, OPCAT will present a table at the **Bottom Grid**, labeled **Exposed Things List**, showing the available exposed things, indicating for each thing its **ID**, **Name**, Thing Description **Exposure Information**, which is optionally provided by the exposing user, a **Public** and **Private** checkboxes, the **Model Name** – the name of the model which is the source of the exposed thing, and a **Model Repository Path** –the complete path from the enterprise server of the model which is the source of the exposed thing.

If you are off-line, the **Exposed Things List** will include only the **Exposed** things that are **private** for this model.

Note that the **Exposed Things List** is available for new models only after you saved the model into one of your **Working Copy** directories and marked it to be added to the repository.

ACTION SUMMARY

Press the Exposed icon

3. Marking a Thing as Environmental

As explained in detail below, you can change any **exposed thing** you use. If you just want to use the exposed thing "as is" without making any changes, you should mark it as **environmental**. This may be the case if you are reading for example from a specific file which is not changeable. Note that Reusing a public expose thing and marking it environmental is just like copying a thing. It is merely an appearance of the thing defined in the other model.

4. Exposing a Reused Thing

In many cases you would want to reuse a parent thing from another model and then change it to become your own parent thing. This may be the case, if you want to add things to the original parent thing and then reuse the expanded parent thing. In such cases, make a thing successor and then turn this into a parent thing.

OPCAT allows you to turn any reused thing into a parent by exposing it again, and then using it and exposing it over and over again.

This may become handy, in many programming languages where you first need to "declare" the things you are going to reuse before reusing it. If you are declaring a Parent thing, from another model, we recommend that you reuse the Publicly Exposed thing at your declaration OPD and then turn it into Privately Exposed thing. By doing this you improve the clarity of your model.

In addition, note that functions related to Public things are available only when you are on-line. Private things may be used when on-line or off-line. If you are planning to work off-line, make sure that you have turned to Private any Public thing you plan to reuse when designing off-line.

5. Reports on Dependencies of Programs on Reused Things

When reusing an exposed thing it is important to know the dependencies between the exposed thing and programs that are already reusing it. OPCAT includes several reporting mechanisms to help you identify such dependencies. The reports are available from the **Exposed Things List** or by right clicking on an **Exposed** thing.

5.1 Dependency Reports from the Exposed Things List

5.1.1 Local Successors Report

To see all the successors of a **Publicly Exposed** or a **Privately Exposed** thing in your model, right-click on the thing in the **Exposed Things List**, select **Reports** and then **Local Successors**. A new tab, labeled **Local Successors,** will be opened in the **grid** with all the successors of this thing in your model. Double clicking on a successor will take you to the OPD with that successor.

ACTION SUMMARY

Select the **Exposed Things List** icon at the top tool bar → right click on a thing → Reports → Local Appearances

5.1.2 Show Successors Report

In some cases you may want to see the global list of enterprise-wide OPCAT models in which some **Publicly Exposed** thing is used. This may be useful when considering alternative **Publicly Exposed** things to be selected for reuse in your model. To see this list of enterprise-wide OPCAT models in which some **Publicly Exposed** thing is used, right-click on the thing at the **Exposed Things List** and then select **Reports** and **Show Successors**.

ACTION SUMMARY

Select the Exposed Things List icon at the top toolbar → right click on a thing → Reports → Show Successors

5.2 Dependency Reports by Right-Clicking a Thing

5.2.1 Show Local Successors (Right-Clicking on a Used Thing)

When you want to know about all the successors of an **exposed** thing in your model and where each one of them is used, right click on the thing marked as used , select **Expose** and then **Show Local Successors**. A new tab will be opened showing all the successors of this parent thing in

your model. The grid includes the **ID** of the thing, the **Current Model Name** which is the name of this thing in the current model, **OPD Name**, **Source Model Name** which is the name of this thing in the model where it was exposed and whether the thing is private or not. Note that the Current Model Name and Source Model Name will be identical unless the name was changed in your model. You can click on any successor and OPCAT will take you to the OPD where it appears.

ACTION SUMMARY

Right click on a thing → Expose → Show Successors

5.2.2 Show Successors (Right-Clicking on an Exposed Thing)

When you want to know about all the successors of an **exposed** thing in all the models in the repository, right click on the thing marked as exposed, select **Expose** and then **Show Successors**. A new tab labeled **Thing Name Successors** will be opened showing all the successors of this thing in your model and in other models.

ACTION SUMMARY

Right click on an exposed thing → Expose → Show Successors

5.2.3 Show Parent Thing

When you expose a thing privately and then reusing it several times in your model, it is helpful to see the parent thing i.e where this Privately Exposed thing was originally exposed. To find this location in your model, right click on a Used Private thing, select Expose and then Show Parent Thing. You will automatically jump to the OPD where the parent thing was exposed.

ACTION SUMMARY

Right click on a Used Private thing → Expose → Show Parent

5.2.4 Open Parent Model

In many cases you would want to review the model in which a publically used thing was exposed. To see the model where a Publicly Used thing was defined, right click on a Publicly Used thing, select Expose and then Open Parent Model. A new OPCAT successor will be opened showing the source model. Note that the model is located in a temp directory and is not checked out. Do not attempt to make changes to this model!

ACTION SUMMARY

Right click on a Use Public thing >Expose>Open Parent Model

6. Expose Options and Symbols

There is variety of exposure situations in which a thing can be. In many cases, after you use a thing you would want to expose this very thing again. This may be the case if you added things to the used thing and now want to make the original and new things available for future reuse.

The possible exposure situations and corresponding labels are presented in Table 1. Originally, the label is green. If it is changed, its color will be red.

Table 1. Exposure Labels

	Thing Exposure Situation	Exposure Label
1.	Publicly exposed	E-PB
2.	Privately exposed	E-PR
3.	Publicly and Privately exposed	E-PB-PR
4.	Use-of- Privately exposed thing which was then, Publicly and Privately exposed	UPR-EPB-EPR
5.	Use-of- Privately exposed which was then Publicly exposed	UPR-EPB
6.	Use-of- Privately exposed thing which was then Privately exposed again	UPR-EPR
7.	Use-of-Privately exposed thing	UPR
8.	Use-of-Publicly exposed thing which was then Publicly and Privately exposed	UPB-EPB-EPR
9.	Use-of- Publicly exposed thing which was then, Publicly exposed again	UPB-EPB
10.	Use-of- Publicly exposed thing which was then Privately exposed	UPB-EPR
11.	Use-of- Publicly exposed thing	UPB
12.	Use-of-a thing originated from a Template* which was then Publicly and Privately exposed	UT-EPR-EPB
13.	Use-of-a thing originated from a Template* which was then Publicly exposed	UT-EPB
14.	Use-of-a thing originated from a Template* which was then Privately exposed	UT-EPR
15.	Use-of-thing from a Template*	UT

* See the next chapter for Explanation about Templates

7. Exposing a Thing

As noted, a thing (process or object) can be defined and used in the model you are working on, in which case it is Private, or it can be defined or used in another model and in such case will be referred to as

Public. By exposing a thing you declare that this thing is suitable for re-use. It means that other programs may use this thing.

To expose a thing (object or process), simply right click on it and choose Expose. You can now select whether you want to expose this thing publicly (for public use in other models) or privately (for private use in this model). You may expose a thing for both public and private use. This may be handy if you plan to continue and model when you are off-line. A corresponding label (see Table 1) will appear after exposing. You may need to click anywhere in order for the label to appear. Privately Expose things will be added to the Exposed Things List immediately. Publicly Exposed things will be added to the Exposed Things List the next time you commit the model to the repository.

Before exposing a thing, make sure it is ready for exposure. For a **process**, make sure that all the objects which are needed for its operation exist in their proper state. Those things, which are often referred to as the programs API (Application Program Interface), will be validated later when used in other models. Note that when exposing an **object** you actually expose its entire structure. This means that all its parts and attribute objects can be used by other models together with the parent object.

ACTION SUMMARY

Right-click the things → select Publicly or Privately Expose → save model → commit

8. Exposing Things while Being Off-line

When you are working off-line, only Privately Exposed Things will be available to reuse. You may also expose things privately while being offline. Note that for Privately Exposes things to work, the file must be saved in one of your Working Copy directories and marked as Added even if it has not yet been committed. If you are working off-line, remember to save the file in this location and perform Add before you try to expose any thing in the model.

ACTION SUMMARY

New file when off-line → save the file in a Working Copy directory → in the Models Tab right-click on the file → Select Add → Right-click the things>select Expose Privately

If the file is already saved in this location and added then just Right-click the things → select Expose Privately

9. Reusing a Thing

9.1 Selecting and Using an Exposed Thing

To use an expose thing, select the exposed thing you would like to use from the Exposed Things List according to it's name and description. After selecting the right thing you want to use you can now add it to your model by clicking Use at the bottom grid, or right click on the thing in the grid and select Use. If you want to add a thing inside an in-zoomed process, mark the process at the in-zoom OPD before choosing **Use**. You can use as many successors of the thing by clicking

Use again. After Using a thing Save the model in order for your selection to be registered. Note that Used processes can not be in-zoomed or unfolded. Their way of operation is defined by the exposing successor.

ACTION SUMMARY

Bottom Grid> mark a line > press USE

Bottom Grid> right click on a line> select USE

9.2 Setting the Interface or Structure of a Used thing

As mentioned, programs, tables, routines and any other subsystem or procedure have an eco-system in which they can operate and be reused. The required interface or available structure is determined by the model of the exposed thing. When reusing a Publicly or Privately Exposed thing you must make sure that you do it correctly.

In order to ensure that a Publicly or Privately exposed thing is reused correctly, its interfaces or structure must be in line with Exposed thing. As explained below, this is exactly what OPCAT's **Interface Advisor** is designed to do. Note that after the parts or interface defined by the exposed thing are in place, you may add your own parts or additional interfaces. This may be handy when defining tables. It is less practical when using a program API where interfaces you add would probably be ignored.

9.2.1 Connected Vs. Disconnected Interface or Structure

An exposed thing's interface or structure is defined in its source model. When using the exposed thing you may want to register that the things you are using are successors of the one appear in the source model. In such case, when using the Advisor select **Add Connected**. If you would like to define your own interface or structure, but use the once exists at the source model as draft, select **Add Not Connected**. In case you are not sure, we recommend adding things as connected.

9.2.2 Setting a Reused Process Interface with interface Advisor

The **Interface Advisor** helps you set the reused process interface. Always add interface things to a reused process via the **Interface Advisor**! After fetching an Exposed process via Reuse, right-click on the process and select Interface Advisor. The Advisor will then present at the bottom grid the list of missing interface things (which you can think of as missing parameters) and suggest adding them in order to complete the interface. You must select them one by one. You can then select the Interface Advisor again. Each interface item which was added is marked at the list. Continue this operation until there are no more unmarked things in the list. You may select to add each interface item as either connected or not connected Note that you must be in the OPD where you would like to add the interface.

Interface Advisor works for Processes as well as for objects, although for objects it may be less practical.

The **Interface Advisor** works for Publicly and Privately Used things. The **Interface Advisor** for Publicly Used things is available only when you are online and have logged in to the **Repository**. If the **Interface Advisor** appears empty for Publicly Used things, make sure to check that you are online and logged in.

ACTION SUMMARY

Right-click on a Used process → select Interface Advisor → at the bottom grid select the thing you want to add → click Add Connected or Add Disconnected

9.2.3 Reusing Parts and Attributes of a Reused Object with Properties Advisor

An object may be composed of or defined by one or more other reused objects which may in turn be composed of or defined by many other reused objects. Each object can be defined in a separate model. This means that the structure of an object may be aggregated from several places. In OPM terms this may also be the case for processes. Nevertheless, the case for processes is less practical at this stage and therefore is not enabled in OPCAT 4.0.

Exposing an object in OPCAT makes all its children available for reuse. To do this select a Publicly or Privately Reused object and select **Properties Advisor**. The child objects of the Reused object will be presented at the bottom grid. Select one by one only the ones you are going to use in your model. You can add each new child object as either **connected** or **disconnected**. If you would like to use the definitions of the child object you are adding via the **Properties Advisor**, add the thing as **connected**. Note that Properties does not include things that inherit from the Exposed object (i.e connected with Generalization Specialization relation). Also note that you must be in the OPD where you would like to add the property.

ACTION SUMMARY

Right-click on a Reused process → select Properties Advisor → at the bottom grid select the thing you want to add → click Add connected or Add not-connected

10. Changing a Reused Thing

Reuse of an Exposed Thing by other programs creates a dependency between the models or programs (in the case of public reuse) or between different parts of the same model or program (in the case of private reuse). Therefore, the changing an exposed thing that is already being reused must be carefully managed. After an Exposed thing is reused by other models, it can no longer be changed without taking into consideration that others reuse it.

According to this principle, once an exposed thing has been reused by other models, you cannot change it until all the users of the exposed thing have "released" it. You may make local changes, but you will not be able to commit your changes until everyone else have released this exposed thing.

10.1 Disabled Changes to a Used Thing

The following changes to an exposed thing which is reused are disabled until the thing has been released:

- Processes – any change to procedural links connected to this process.
- Objects – any change to structural relations connected to this object, including addition of new parts and attributes, and any change to the type of the object.
- Things – Deleting an exposed thing or its interface is disabled (even locally) until that thing has been released.

10.2 Enabled Changes to a Reused Exposed Thing

You can make the following changes to an Exposed thing without the need to release it:

- Processes – Change the way an Exposed process operates, as specified by its sub-processes. You should still notify other users, as explained below. However, OPCAT is not preventing you from committing such changes.
- Objects – Change procedural links to an Exposed object. For example, if the object **Table A** is exposed and reused, and it is updated by the process **A Updating** (the two things are connected with an Effect Link), you may remove this link and link Table A to another process, **New A Updating** without the need to release **Table A**.
- Things – Update the information presented to other users who consider reusing this Exposed thing. To do this, right-click on the thing, select Expose and then select Update Exposure Information.
- Things – Change the name of a thing or its description.
- Appearances – if you created few appearances for the same Publically or Privately exposed thing by using **Copy**, you may delete the appearances except for the original appearance from which you copied.

10.3 Changing an Exposed Thing

If you want to change an exposed thing for which release is required, follow the steps described in the next sections.

10.3.1 Make Changes Locally

First, make sure you committed any other changes you made before changing the exposed thing. Make changes to the exposed thing and save them locally. Once you are done changing the exposed thing you will need it to be released by the other models in which the thing is reused before you will be able to commit the model.

10.3.2 Request Release

Check who is reusing the Exposed thing you need to be released by clicking the Exposed Things List icon, right-clicking on the Exposed thing you changed and select Reports → Show Successors or Local Successors reports as applicable. You can then see if there is any local or global reuse of this thing. If the thing is not reused anywhere, you can proceed to commit the model and skip the rest of the process. If there is only private reuse (reuse in your own model) you need to release the exposed thing in you model, as explained in the next section. If other models are reusing this thing, then you need the Exposed thing to be released at each model in which it is used.

You can notify other users of the expected change by using **OPCAT Messages**, or any other organizational communication means.

To use **OPCAT Messages**, right-click on the exposed thing in the Exposed Things List and select **Messages**. You are now presented with two options: Interface Needs Changes or Non-Interface Changes. Only interface changes require that the Exposed thing be released. However, you should notify other users also of non-interface changes, although OPCAT does not prevent you from committing the changes without getting other users to release the Exposed thing.

Use the Interfaces Needs Changes option to notify users that they are requested to release the exposed thing, so you can commit the changes to the interface. Every user that opens OPCAT will now see the messages at the Messages Console. You can send a message requesting to release an Exposed thing even if you are not currently working on the model that contains the Exposed thing. You can also send a message to specific model. To do this, right-click on a thing in the Exposed Things List and select Reports, then right-click again on the model which you want to send message to and click Messages.

10.3.3 Releasing an Exposed Thing (by the Exposer)

This section is intended for users who use a Publicly Exposed thing which is about to be changed or use a Privately Exposed thing which needs to be changed in the model.

If you are using Publicly Exposed things, check the Messages Console for information about planned changes. At the Message Console, you will see messages about things that were exposed and un-exposed.

To release an Exposed thing, right-click the Exposed thing, select Expose and then select Release Publicly Exposed or Release Privately Expose, as applicable. Note that no roll-back is available for this action. Now save, and if the thing was Publically Released, then also commit the model. Once this is done the Exposed thing can be changed.

ACTION SUMMARY

Right-click on a reused thing → select Expose → Release Publicly Exposed or Release Privately Exposed→ Save→ Commit

10.3.4 Commit the Changes to Exposed Thing (by the author)

In order to know whether a thing was released by the owners of all the models in which that thing is privately or publicly reused, you can check the Message Console or the reports available by right clicking on the thing in the Exposed Things List. When the Exposed thing is released by the owners, you can commit the model. Note that you can not delete an Exposed thing or interface of Exposed thing even locally until it is released. Therefore, if your changes include deletion of exposed thing or interface of an exposed thing you should now open the model and delete the items before committing.

10.3.5 Reuse the Modified Exposed Thing Again

When you **Release** a thing, you allow the owner of the model where the **Parent** thing resides to change it. The thing itself will remain in your model, it but will no longer be connected to the exposed thing. However, that thing's interface will remain in your model as is, i.e., if any part of it was connected, it will remain connected even though the thing itself became disconnected.

Once the exposed thing was changed by its owner, you can reuse it again. For the sake of caution, once a thing was released, it can no longer be "re-reused". The only way to make a released thing reused again is to add it again from the Exposed Things List. After adding the Exposed thing at the right place, use the Interface or Properties Advisor to reconnect the Exposed thing to its interface. The Advisor is working according to the changed interface or structure currently exists for the exposed thing, so any changes made since you released the thing will be recommended by the Advisor. After completing this, you can delete the previous Exposed thing and interface things which are no longer linked.

ACTION SUMMARY

Click the Exposed Things List → Select the thing you want to use → press Use → Right click on the thing in the model → select Advisor → add the interfaces → delete the original thing and interface

10.4 SAVE AS

"Save As" seems like a simple operation. Yet, when your model contains **Publically Exposed** things this concept becomes a bit more complicated. When selecting **Save As** for a model that includes reuse, the Privately Exposed and Privately Used things remain intact. Publicly Used things also remain without any change although the Successors List will be updated automatically to accommodate the use of the Exposed thing by the newly created model. The Publically Exposed things from the original model will be automatically Un-Exposed Publicly. Any usage of those Publicly Exposed things will remain with the original model.

Templates

Templates are the organization's best practices for modeling frequently used systems, modules, programs or expressions. Using templates ensures compliance with a specific way of design which becomes the organization's policy or guidelines. Templates that were prepared correctly and thoroughly tested can save you plenty of time. Only people with special expert privileges can save templates to the reserved Templates directory in your organizational repository.

1. Principles for Creating an Organizational Template

When you create a template, remember that it should serve as a skeleton for different models. The template model should be as simple as possible. Add only things which are relevant as a skeleton. Do not add things which are specific, unless the template is designed to handle this specific situation. Do not in-zoom unless this is necessary. Make sure that the template is compatible with OPM rules.

Templates are divided into two types: **Start-off templates** and **Constructs**. A **Start-off template** provides you with a ready-made model to start from. A **Construct** determines how to model a specific operation or structure in OPM. If you are preparing a Construct Template of a process, make an effort that the process you are modeling exists in one OPD only. If you have the necessary privileges, once your Start off template or construct template model is ready, all you need to do is to commit the model to the Templates directory.

2. Starting a New Model from a Template

To start a new model using a template simply select System → New. You will be presented with a pop-up window asking you to select **New** or **From Template**. Select **From Template**. A drop-down menu appears, listing all the organizational templates that are available for you to use. Select a template and click OK. You can now edit the template locally. The template model file is saved in a temporary library so before you start editing it, it is recommended that you save it in your local **Working Copy** environment using **Save As**.

ACTION SUMMARY

System → New → select From Template → OK → Save As

3. Using Constructs in an Existing Model

If you design a program or model a system that includes a **Construct** such as an operation or a data structure which is commonly used in your organization, such a **Construct** should be made available in the **Templates** directory. You can then use this **Construct** in your model and validate that you used it correctly. To use a **Construct**, open the **Templates** directory in your **Working Copy** environment. This

directory is automatically synchronized with the server and includes the most updated **Templates** and **Constructs**.

To use a **Construct** from a model, click the **Models** tab on the left pane, right-click on the model which contains the thing you want to have as a construct, and choose **Add as Template**. The model is now added into the **Templates** tab. Click the **Templates** tab, then click on the key icon next to the model containing the thing in which you are interested. You will see all the things in the model. Right-click the thing you want to use as a construct in your model, and from the pop-up menu choose **Insert**. The thing you have chosen is now added to your model. If you want to add a thing inside an in-zoomed process, mark the process at the in-zoom OPD before choosing **Insert**.

You can use any model which exists in the repository as a **Construct**. In order to do so, right click on the model in **Models** tab and select **Add Template**. The selected model will appear in the **Template**s tab. This model will be available as template only if the template exists in the **Working Copy** in the computer where you work on the model. If you are using a different computer you must check out the local template as well. It is therefore recommended that you do not add Constructs that are not located in your Working Copy and synchronized with the repository.

ACTION SUMMARY

Models tab → open Templates directory → right click a model → select Add as Template → go to Templates tab → press the key → chose the construct you want to use → right-click and select Insert

4. Adding the Entire Construct

Just like using an Exposed thing, when using a Construct you would like to make sure that the entire Construct was added. In order to do this use the Interface Advisor (for processes) and Properties Advisor (for objects) as explained in the section "Setting Interface or Structure of Reused Thing" above. The Advisor will automatically refer to the template model as the source model.

ACTION SUMMARY

See "Setting the Interface or Structure of a Used Thing" above

Troubleshooting

1. While trying to commit, the File Console notifies "transmitting file…" but does not complete the action:

   In most cases the reason is that the directory or file was deleted from the repository after it had been checked out to your local directory. To verify if this is the reason, go to the Admin Console where it should say "Path does not exist". If you still want to check this file or directory into the repository you can create a directory with the same name at the repository, check it out to your local copy and then copy the files to the newly created directory in your local copy. Then you will need to add the files and commit them again.

2. I exposed a thing, but it does not appear in the Exposed things list.

   Make sure that you committed the file to the directory after exposing it. If this does not help, verify that you expose it for public use.

3. I am trying to reuse an exposed thing, but when I click **Use** I get "Cannot Insert Exposed Thing into selected Instance" error message.

   Check that no object or process is selected in your model. If it is selected, the **Use** mechanism thinks you are trying to add something inside the selected thing. which is not allowed in OPCAT. To solve this, unselect the thing and press **Use** again

4. I cannot delete files from the repository by **Delete File** in my **Models.**

   Make sure that the files are not locked by other user.

5. I cannot commit a file to the Repository.

   There can be several reasons for this:

   a. Check that you are on-line.

   b. Make sure that the file is not locked by another user.

   c. Check the Admin Console for more information about the reason for failure to commit the file.

6. Why does the advisor show an empty tab for a Publicly Exposed thing?

   The Advisor for Publicly Exposed thing is reading from the repository. In order for it to work you must be online and logged-in.

7. I can not find an exposed thing in the exposed things list?

   a. If you can not find a thing in the Exposed Things List for example check that the System Name is set as you expect it to be.