Volume

2

OPCAT SYSTEMS

iVision

User Guide

iVision

OPCAT SYSTEMS

iVision User Guide

Table of Contents

Chapter 2

Purpose of this Guide

The purpose of this guide is to allow experienced users of OPCAT 3.0 to work with the new features offered by iVision, which include transition to server architecture, the use of OPCAT Model Control (OMC), cross-system dependencies design and tracking mechanism, categorization, versioning and much more. iVision suite includes a Server, a design tool called OPS-Object Process Studio and a browser based reporting module. Readers with no experience in OPM who are interested in learning how to model are referred to the *Modeling with OPCAT* tutorial, which can be found on OPCAT website www.opcat.com. Administrators should also read iVision Administrator Guide. Help for iVision reporting is available on-line after entering iVision web-based reporting. Readers who are designing RPG code should also read the *RPG Conversion Guide*. Finally, if you wish to perform real model-driven design of code in your organization, you are advised to read *iVision Step-by- Step Model Driven Design Process* presentation.

What's New in iVision?

iVision is designed for use by enterprises. To this end, it includes a Server which runs OPCAT Model Control (OMC) and messaging system modules. Each OPS client is communicating with this server. OPCAT has been further enhanced to allow non-technical users to understand and manage their teams and systems. This is done via iVision web-based Reporting module. Additional features, including Exposing and Templates, were added to enable cross-system reuse-enhanced modeling and model management. The new Categorization module allows users to classify model elements according to organizational classification criteria.

OPCAT Model Control (OMC)

1. Concepts

OPCAT Model Control (OMC) is the information sharing module of iVision. 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. Users may approach the repository using Object-Process Studio (OPS) – iVision's client.

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 local **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 working 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

OPS 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 had 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 OPS's Repository Browser tab or by performing any action involving the repository, a login screen will appear, requiring your credentials.

Note that different users may use the same computer. In this case, make sure you check out the files you are working on before committing, even if those files already exist at the Working Copy in order to clean-up the credentials.

2. User Consoles Shown in OPS

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 to "information" for all messages). Here you will also find messages related to changes to Exposed things, as explained in detail in the sequel.

3. Basic Controller

The left pane in OPS 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.

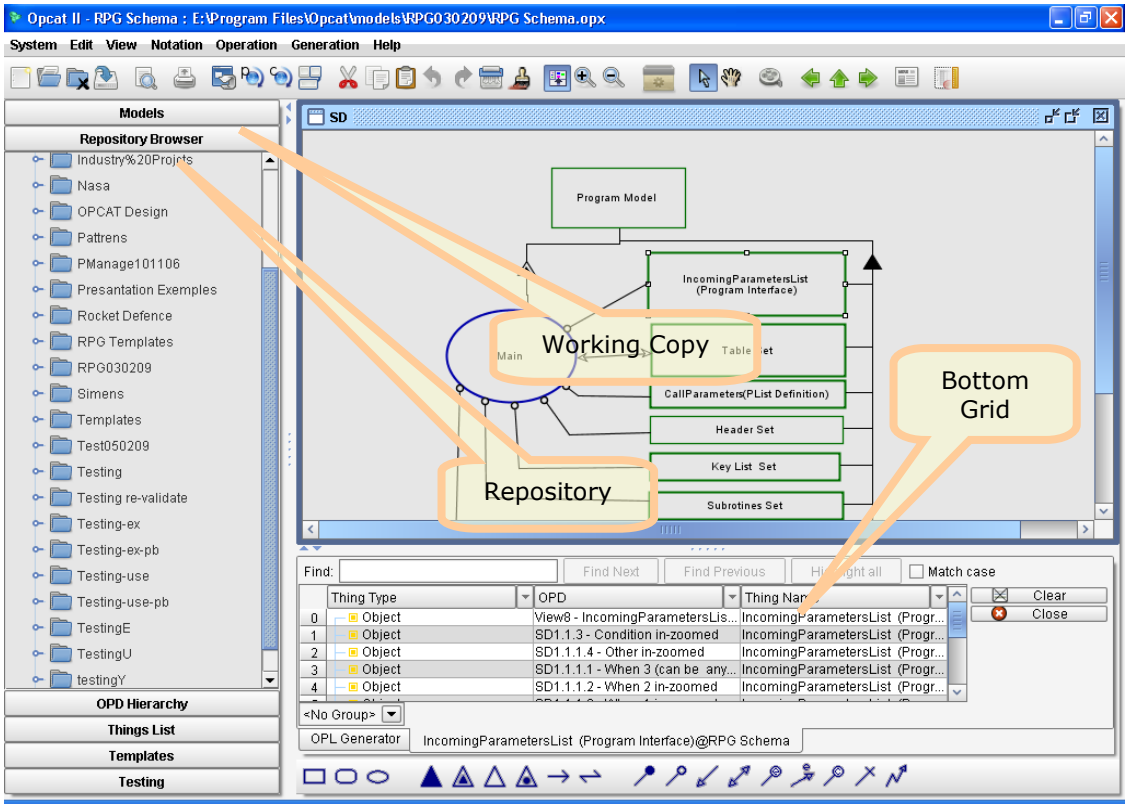

Figure 1. The Working Copy, Repository, and Bottom Grid

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 directories or single files. 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.

You may also check out a single file. To do so, right-click on the file at the **Repository Browser** and select **Checkout.** Checked out files will be added at the right place in your Working Copy environment. In case the directory was not previously checked out, it will be automatically created in your Working Copy environment, and the file will be automatically placed at this directory.

Action Summary

Repository Browser → Right-click directory or file → 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 on the file icon 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 Get

Get does the same thing as Checkout but without locking the file. A **Get** command will copy the directory with all its files from the Repository to your local folder. A checked-out file will **not** appear as **locked** for other users. If you want to lock the directory or file you got later on, you may use the **Lock** command at explained below.

Action Summary

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

4.4 Import

Import allows you to add files or directories which have not yet been added to the repository 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

4.5 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, 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 wish to undelete a file, 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

4.6 Revisions

Not yet implemented.

4.7 Show Properties

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

ACTION SUMMARY

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

4.8 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

4.9 Add Directory

You can create a new directory in your repository by using **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

4.10 Model Symbols in the Repository Browser

Figure 2**Error! Reference source not found.** 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: **Models** and **Working Copy**. The **Models** folder 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 OPCAT session. OPCAT cannot open more than one model in one session, 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 session 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 editing 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 pop-up 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 over and over 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 back files which exist in the repository but are missing from your **Working Copy** directory. **Update** works just like **Checkout** for the missing files. **Update** for a single file is visible in the menu, but it will be operational only in an upcoming 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 retrieved 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 the 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, right-click and select **Cleanup**.

ACTION SUMMARY

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

5.12 Add or Make Directory

See Section 5.4 *Adding new Directories or new Files from your local drive to the Repository*.

5.13 Delete Locally

Use **Delete Locally** when there is a file or directory on your local working copy which you no longer need. This may be the case if you committed one or more files or directories, you unlocked it, and you do not plan to continue working on it, but it is still part of your local copy. In this case, you should delete those files or directories from your working copy. If you later need one of these files or directories 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 Read-Only Models

If you check out a model which was locked by another user, you will get it as read-only. The model will be marked with a small lock icon in your Models tab. If the model was then unlocked by the other user, you may now check them out for editing. Before doing so, **Delete Locally** the file 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

5.15 Add Local Directory to the Models Tab

You can add one or more directories to your "Local Models" root if you want to use them to browse and open models on your local machine, but these directories will not be added to the repository. Only directories under your Working Copy are managed by the OMC and can be added to the repository. To add a local directory to **Local** 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

5.16 File Names

You can provide names to your models as you see fit according to your organization's conventions. Names can be in any language but cannot contain the @ symbol. Note that each model has both a File Name and a System Name. The File Name appears at the ribbon on the top of the OPD window. The System Name appears as the root of the OPD Hierarchy tab on the left. The File Name and System Name need not be identical. OPCAT always references the System Name.

5.17 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 yet committed.
	The file is scheduled to be deleted upon the 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. To increase productivity and to save time and cost, whenever possible, you will be required to use existing code in programs or systems you create. However, reusing the same things multiple times creates a web of cross interdependencies between different parts of systems and code. Reuse must therefore be done wisely. If you don't know what programs are available to you for potential reuse, it is less likely that you will choose the right program. Conversely, if you reuse some other program and the developer in charge of that program is not aware of this reuse and therefore changes the program, this can lead to dire unintended consequences.

iVision is designed with reuse and code and systems unification in mind while reducing potential reuse pitfalls. To this end, iVision provides the designer with a list of all the available items—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 for reuse, called the **Exposed Thing**.

6. Reuse: Basic Definitions

Before discussing how to expose and reuse things, we explain the underlying concepts related to the existence of an item in several models or in several places within a model. 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 in the same system, 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 active when one of them does.

In order to reuse things, iVision 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.

6.1 Thing Parent and Successor

An **object parent** is an abstract collection of objects, for which the set of features (attributes and operations) and their permissible, legal states and/or values is defined.

When you **Expose** an object, you turn this object into an **object parent**.

An **object successor** is a uniquely identifiable object 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 you **Reuse** an object, you create an object successor which is derived from the object parent, i.e., the exposed object.

Different object successors originating from the same parent have the same type as the parent. They have different identifiers and may have different names, values and/or states at the same point in time. 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, for which the set of features (attributes and operations) and their permissible, legal states and/or values is defined.

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.

A **thing parent** is an **object parent** or a **process parent**.

A **thing successor** is an **object successor** or a **process successor**.

A **source model** of a **thing successor** is the model in which its **thing parent** exists.

6.2 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, a certain 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 **thing** that may appear any number of times in various OPDs in the same OPM model, where **thing** in this context is 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.

In OPCAT, you create thing appearances by a simple copy-paste operation on the thing for which you wish to create a new appearance.

6.3 Private and Public

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

7. The Exposed Things List

All the exposed things in your organization are presented in the **Exposed Things List**. To view the **Exposed Things List**, click on the **Exposed** icon, which is located at the top toolbar and looks like this:  . In response, OPCAT will present a table at the **Grid** in the bottom panel, labeled **Exposed Things List**, showing the available exposed things and indicating for each thing its **ID**, **Name**, Description, **Exposure Information**, which is optionally provided by the exposing user, a **Public** and **Private**, the **Model Name** – the name of the model from which the **Exposed Thing** originated, and a **Model Repository Path** – the complete path from the enterprise server of the **Exposed Thing**'s **Source Model** – the model which is the source of the **Exposed Thing**.

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. 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**. If you are working off-line, the **Exposed Things List** includes only the **Exposed Things** that are **private**, i.e., are exposed only for the model you are currently working on.

ACTION SUMMARY

Press the Exposed icon

8. Marking a Thing as Environmental

As explained below, you can change any **Exposed Thing** you use. If you just want to use the exposed thing "as is," without making any changes to it, you should mark it as **environmental**. This may be the case if, for example, you are reading from a specific file which is not changeable. Reusing a Publicly Exposed Thing and marking it as environmental is the same as copying and pasting that thing. In other words, it is merely yet another appearance of the **Exposed Thing** defined in the **Source Model**.

9. Exposing a Reused Thing to Make It a Parent Thing

It is often the case that you would want to reuse an **Exposed Thing** (object or process) from another model and then change it, making it your own parent thing. For example, you may want to add parts or attributes to the original parent thing and then reuse the expanded parent thing. To do this, you need to turn a successor thing into a new parent thing. OPCAT allows you to turn any reused thing into a new parent by exposing it, and this can be repeated over and over again.

This is similar to declarations in many programming languages. If you declare 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.

Functions related to **Public Things** are available only when you are online. **Privately Exposed Things** may be reused even if you are off-line. Therefore, if you plan to work off-line, first make sure to turn any **Public** thing you plan to reuse while working off-line to **Private**.

10. Reports on Dependencies of Programs or Models 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 either from the **Exposed Things List** or by right-clicking on an **Exposed** thing.

10.1 Dependency Reports from the Exposed Things List

10.1.1 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 Successors

10.1.2 Show Successors Report

In some cases you may want to inspect 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, right-click on the thing at the **Exposed Things List**, 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

10.2 Dependency Reports Produced by Right-Clicking a Thing

10.2.1 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 reused, 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

10.2.2 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

10.2.3 Show Parent Thing

When you expose a thing privately and then reuse it several times in your model, it is helpful to see where the Privately Exposed thing was originally exposed. To find this location in your model, right click on a privately used 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

10.2.4 Open Parent Model

In many cases you would like to inspect the model in which a publicly used thing was exposed. To do this, right click on a Publicly Used thing, select Expose and then Open Parent Model. A new OPCAT session 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

11. Expose Options and Symbols

A thing can be in various exposure situations. After you reuse a thing, you may want to expose this thing again. This may be the case if you added successor things to the reused thing and now want to make the original reused and new successor things available for future reuse.

The possible exposure situations and corresponding labels are presented in **Error! Reference source not found.**. 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 thing which was then Publicly exposed	UPR-EPB
6.	Use-of-Privately exposed thing which was then Privately exposed	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 originating 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 next chapter for Explanation about Templates

12. 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 Privately Exposed, or it can be defined or used in another model, in which case it is Publicly Exposed. By exposing a thing, you declare that this thing is suitable for reuse by the same system model (in case of private exposure) or by other system models (in case of public exposure).

To expose a thing, 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 modeling off-line. A corresponding label will appear after its 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 signature or the program's 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

13. Exposing Things while Being Off-line

When you work offline, only Privately Exposed Things are available for 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 offline, remember to save the file in this location and perform *Add* before you try to expose anything 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

14. Reusing a Thing

14.1 Selecting and Using an Exposed Thing

To reuse an exposed thing, select the exposed thing you wish to reuse from the Exposed Things List according to its name and description. After selecting the thing you 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 OPD in which the process is in-zoomed before choosing **Use**. You can use as many successors of the thing as you like by clicking Use again. After Using a thing, Save the model in order for your selection to be registered. Note that reused things cannot be in-zoomed or unfolded, as they are constrained by their parent thing.

ACTION SUMMARY

Bottom Grid > mark a row > Use

Bottom Grid > right-click on a row > select Use

14.2 Setting the Interface or Structure of a Reused 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 may not be usable by the program you have used without changing its internals.

14.3 Connected Vs. Disconnected Interface or Structure

An exposed thing's signature (interface and/or structure) is defined in its source model. When using the exposed thing, you may want to enforce this signature. To do this, when using the Advisor, select **Add Connected**. If you would like to define your own signature using the one from the source model as draft, select **Add Not Connected.** In case you are not sure, we recommend adding the thing as connected.

14.4 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 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. The Interface Advisor works for Processes as it does 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 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

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

A thing may be composed of or defined by one or more other reused things which, in turn, may be composed of or defined by other reused things. Each thing can be defined in a separate model. This means that the thing may be combined of things from several models. Since the case for processes is less practical, it is not enabled in iVision.

Exposing an object in OPCAT makes all its successors available for reuse. To do this, select a Publicly or Privately Reused object and select **Properties Advisor**. The successor objects of the Reused object will be presented at the bottom grid. Select only the ones you are going to use in your model. You can add each new successor object as either **connected** or **not connected**. 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 are inherited from the Exposed object (i.e., connected with a 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

15. 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, 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 the possible impact on other models that reuse it.

According to this principle, once an exposed thing has been reused by another model, 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.

15.1 Disabled Changes to a Used Thing

The following changes to a **Publicly Exposed** thing which is reused are disabled, i.e., the model's file cannot be committed, until the thing has been released:

- For a process – any change to a procedural link connected to this process.
- For an object – any change to a structural relation connected to this object, including addition of new parts and attributes, and any change to the type of the object.

- Thing – Deleting an exposed thing or its interface is disabled (even locally) until that thing has been released.

Except for deletion and **Private Un-Expose**, OPCAT will allow you to make the above-mentioned changes to **Privately Exposed** things that are reused after providing a warning message. Deletion of **Privately Exposed** things is disabled. **Private Un-expose** can be done locally but would prevent you from committing the file until the successor is released.

15.2 Enabled Changes to a Reused Exposed Thing

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

- For a process – Change the way an Exposed process operates, as specified by its sub-processes. You should still notify other users, as explained below. However, OPCAT does not prevent you from committing such changes.

- For an object – Change of a procedural link to an Exposed object. For example, if the object **Table A** is exposed and reused, and it is updated by the process **A Updating**, i.e., 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**.

- For an Exposed thing – 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**.

- For a thing – Change the name of a thing or its description.

- For an appearance – if you created few appearances for the same Publicly or Privately exposed thing by using **Copy**, you may delete all the appearances except for the original appearance.

15.3 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.

15.3.1 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 owners of the other models in which the thing is reused before you will be able to commit the model.

15.3.2 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 the reuse is only private (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 **OPS Messages**, or any other organizational communication means.

To use **OPS Messages**, right-click on the exposed thing in the Exposed Things List and select **Messages**. You are now presented with two options: Interface ~~Needs~~ Change Request 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 a 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.

15.3.3 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

15.3.4 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 cannot delete an Exposed thing or the interface of an 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.

15.3.5 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 reusable 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 works according to the current signature (interface or structure) of 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

15.4 SAVE AS

"Save As" seems like a simple operation, but the consequences of this operation are the creation of an additional version of the same thing, in which all your previous usage and exposure will be eliminated and you will have to recreate the dependencies.

Templates

Templates are the organization's best practices for modeling proven 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 correctly prepared 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.

16. 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: **Startup templates** and **Constructs**. A **Startup 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 sure that the process you plan to use as Construct exists in exactly one OPD. If you have the necessary privileges, once your **Startup** template or **Construct** template model is ready, all you need to do is commit the model to the Templates directory.

17. 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 popup 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

18. Using Constructs in an Existing Model

If you design a program or a model of a system that includes a **Construct**, such as an operation or a data structure, which is commonly used in your organization, that **Construct** has to 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 choose **Insert** from the popup menu. 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 thing in the model which exists in the repository as a **Construct**. In order to do so, right click on the model in the **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 use 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

19. Adding the Entire Construct

Just like using an Exposed thing, when using a Construct you need to make sure that the entire Construct was added. To do this, use the Interface Advisor (for processes) or 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

Categories

The model describes some aspect of our system. There may be additional information we would like to attach to certain model elements. The Categories Module is designed for this purpose. Additional information may be release version for each element, its status (set, modified, to be modified), etc. Another common option is connecting requirements to model elements. The categories and their values are set by your administrator. Any category and value can be set to address the organization's needs. Once the Categories are set and connected, your administrator can view which models are connected to which elements. For example, you can view which source files are going to be changed at the next version.

20. Applying Categories to your model

Once you logged in and opened a model, all the organizational categories are available. You can see the available categories by clicking the Show Categories icon,  , on the top bar. The categories will be opened in tabs at the grid at the bottom of OPS.

20.1 Connecting Model Elements to Categories Values

You can now connect any model element or elements to any Category value or values. This is many-to-many connectivity. To do so, click on the model element you want to connect, then click on the line in the grid in which the Category value appears and then press the *Connect* button found on the right side. Once connected, you will see the model element in the grid at the "things" column.

ACTION SUMMARY

Press  on the top bar → select Category tab → select value by click → select model element by click→ press Connect

20.2 Disconnecting Model element and Category Value

Any user that has read-write authorization for a file may connect and disconnect Category values to model entities. To disconnect a model entity and Category value, select the model entity and then select the line in the grid for the connected Category Value. Now click "Disconnect". Note that disconnecting a Category value is a meaningful event. Therefore, there is no option to disconnect all. Rather, you should think about what you are doing and carefully select the model entity and Category value. Having said that, you may connect or disconnect few model entities found on the same OPD by using the shift key with your mouse. You may also connect few Category values to an entity by using the shift key to multiply select few lines at the grid.

ACTION SUMMARY

Press  on top bar → select Category tab → select value by click → select the model element you want to disconnect by clicking on it → click Disconnect

21. Analyzing relationship between Categories

21.1 Coloring

OPS enables you to understand which Category values are connected to which model entities by coloring the connected model entities according to their value. To color model entities according to a certain Category, right-click on an empty space in your model and then select Meta Coloring. You will now see all the available Categories. Select any of them to color your model according to this Category. If you want to remove the coloring, just select default. Note that the coloring is removed whenever you save your model.

ACTION SUMMARY

Right click on empty space in your model → Meta Coloring → Select Category

21.2 Analyze

Analyze allows you to understand the relationship between different Category values connected to different model entities. When this information is reflected at the model, you can inspect how internal model or program dependencies correlate with external dependencies. For example, you can see that a particular piece of code, which is scheduled for this release, depends on some other piece, which is scheduled only for future releases. To analyze a Category value, press the Categories icon at the top bar. Then, at the grid, select the Category and Category value you would like to analyze. Now, right click on this value and select **Analyze**. A new tab will be opened, listing all the things connected to this Category value, the model elements to which this model elements are connected in terms of OPM, and any Category value to which the other model element is connected. The last columns are dedicated to the OPM link between the two model entities

ACTION SUMMARY

Press  on top bar → select Category tab → right click on a value → select *Analyze*

22. Off-line and on-line work

In principle, you may connect Category values while working online or offline. Note, however, that if a Category was changed by your administrator when you were offline, you will only see the effect of this change when being back online.

23. Changing Categories by Administrators

Categories may be changed using the Admin Console. Explanation about this process can be found in the iVision Administrator Guide. From a user's perspective, you should be aware that the administrator can change the value names, add values and delete values or even an entire category. Before doing so, the administrator is advised to use the Validation tool found at the Admin Console, which informs the administrator which models are connected to any Category value the administrator is about to change. Nevertheless, the administrator may change categories even if they are used in a model. Such changes will be reflected in the model the next time you open the model. If the Category value is deleted, it will then be deleted from your model along with any connections you may have made.

Versioning

24. Model Versions

As mentioned above, each time you commit a model to the server you create a new version. The versions are saved and numbered automatically by OMC. At this section we will describe how to compare and review the differences between two versions of a model. This is handy either before committing a new changed version, or in order to review the changes between the last "good" version and the current version.

Note that when committing a model you are requested to add a commit note. This note will become useful when trying to identify which versions to compare. You may also compare two versions of the same model which are not at the OMC. This may be the result of a "Save As" action. Nevertheless, comparing two models which are not a version of each other is not recommended and would yield unusable results.

25. Selecting Versions to Compare

25.1 Selecting a Model To Analyze

To select a model to analyze go to Models on your left pane and right click on the model you want to analyze. At the menu select *Model Diff*. Note that models will be found under models tab if saved in the Working Copy directory. You do not need to open the model in order to see the differences report. Nevertheless, if you want to review the changes in the model in parallel to the differences report you may want to open the model before selecting *Model Diff*.

ACTION SUMMARY

Model Tab → right click on a file → Model Diff

25.2 Selecting the Versions

After pressing *Model Diff* a selection window will appear. You may select to compare two files on your hard drive and in such case browse for the file by pressing the three dots icon, or you may want to compare two versions at the repository and in such case select the version at the drop down. The version selection at the drop down will be available only if the model is at the repository and you are working on-line. The commit notes and data will appear at the *Commit Remark* window allowing you to select the "right" two versions for comparison.

26. Analyzing Differences

26.1 Diff Results

A grid containing the differences between the models will appear at the bottom pane. At the grid you will find the following tabs (which may have different names if changed by your System Administrator):

Name	the name of element was found different
Type	the type of this element which can be either Object, Process, Link or Relation (Structural Relation)
New, Deleted, Changed	the mark at each of the next three columns shows the essence of the change
State/In-Zoom Change	mark that the change is within the in-zoom in case of a process, or at the state of the object
Appearances Change	appearances are created by copy-paste an element from one place to another for display purposes. This column marks if an appearance was added or deleted (there is no "change of appearance" because a change in one appearance is reflected in all of them)

Note that the same change may yield few rows at the grid. For example if you add a process within an in-zoomed process you will get a row for the parent process which was changed and a row for the new process that was added. In addition you will get a row for relation between that parent and the new process which is not shown on screen but exists at the background.

Colors – New elements are marked in red. Changed elements are marked in blue and deleted elements are marked in red.

26.2 Show Appearances

Double click on the element you want to analyze. This will open a new tab which shows all the appearances of this element. Clicking on an element will take you to the OPD where this element appear

ACTION SUMMARY

Diff Tab → double click on an element → New tab → Click on an appearance

26.3 Show Property Changes (for Changed and Deleted Elements)

At the Diff tab, select the element you would like to analyze by clicking on it once. Now right click and select Diff Actions. You can either Open Parent Model or Show Diff. If you select to

open Parent Model, the other version which is not currently presented will be opened in read only mode in a separate OPS instance. If you select *Show Diff,* a new window will appear. This window contains 3 tabs:

Properties Diff	show the differences in the properties of the element such as name, description, code block, order etc
Appearances Diff	Show new or deleted appearance of an element
State/In-zoom Diff	Show differences in the in-zoom or state properties. The format of the results is: Process name, (OPD:), Order =

ACTION SUMMARY

Diff Tab→ Click on entity → Right click on same entity → select All Diff Actions→ select Show Parent Model

Diff Tab→ Click on entity → Right click on same entity → select All Diff Actions→ Show Diff

Configuring OPS

27. Opcat.properties file

Most of Object Process Studio (OPS) default parameters are set at the opcat.properties file located at the installation directory of OPCAT. Below are the parameters that may be changed by regular users. Other changes may only be done by experienced users. Note that color parameters may be set in RGB or by color name.

Name	Default Value	Description
DBserver	www.opcat.com	Address of your server (database)
MCserver	www.opcat.com	Address of your server (model control)
graphics.default.BackgroundColor	230, 230, 230	Set the background color for objects
graphics.default.BaseBorderColor	black	Set the default color for the main part of the links and relations
graphics.default.LineColor	black	Set the default color for the second part of the links and relations
graphics.default.LineTextColor	black	Color of the text on the links and relations
graphics.default.OPDBackgroundColor	Grey	Set the background color of the model. Will appear when a model is opened
graphics.default.TextColor	black	Set the color for text in the model
graphics.default.ThingIconAlphaComposite	0.75	Set the level of transparency between a thing and an icon which may be attached to it
models_directory	C\:\\Program Files\\Opcat\\Working Copy	This is the location of your working copy. You may want to change this location if you are short of disk space at the default

		location
versions.grid.color.change versions.grid.color.deleted versions.grid.color.new	green red blue	Set the colors that describe version differences

28. Changing OPD Color at Specific Model

Regardless of the default colors selected at your opcat.properties file, you may select different colors for OPDs for specific models. In this case the selected colors will persist with the model. To change the background color for any OPD, right click on empty space at the OPD and select *Background Color*. Next select the color for this OPD.

Troubleshooting and FAQ's

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 that this is the reason, go to the Admin Console, where it should say "Path does not exist". If you still want to check out 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 exposed it for public use.

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

   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 click **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 another user.

5. I cannot commit a file to the Repository.

   There can be several reasons for this:

   a. Check that you are online.

   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 cannot find an exposed thing in the exposed things list.

    a. Check that the System Name is set as you expect it to be.