A User's Guide and Programmer's Manual to Tyche Version 2.2

Handling Simulations with Greater Efficiency and Flexibility

Dave Heppenstall
Contractor

DRDC CORA CR 2007–xx
October 2007

Defence R&D Canada
Centre for Operational Research and Analysis

Maritime Operational Research Team
Centre for Operational Research and Analysis

A User's Guide and Programmer's Manual to Tyche Version 2.2

Running Simulations with Greater Efficiency and Flexibility

Dave Heppenstall
Contractor

Centre for Operational Research and Analysis

Contractor Report

CR 2007-XX

October 2007

Author

D. Heppenstall

Contractor for Maritime Operational Research Team

Endorsed by

A. Jesion

Maritime Operational Research Team Leader

Approved for release by

P. Saunders

Acting Section Head Maritime Operational Research Team

Abstract

<Excised>

Résumé

<Excised>

This page intentionally left blank.

Executive summary

<Excised>

Heppenstall, D. 2007. A User’s Guide and Programmer’s Manual to Tyche Version 2.2: Handling Simulations with Greater Efficiency and Flexibility. [Enter report no.] Centre for Operational Research and Development.

Sommaire

<Excised>

Heppenstall, D. 2007. A User's Guide and Programmer's Manual to Tyche Version 2.2: Handling Simulations with Greater Efficiency and Flexibility. [Enter report no.] Centre for Operational Research and Development.

Table of contents

List of figures

List of tables

This page intentionally left blank.

Acknowledgements

The author is thankful for the continuous support of Cheryl Eisler and Alex Bourque for their support and direction throughout the development and testing of Tyche 2.2.

This page intentionally left blank.

1. Introduction

<Excised>

1.1 Requirements for New Development

It is the classic textbook computer science problem: a complex application is so ingrained within a graphical user interface that it is difficult to separate the basic operations from the user interface. In order to improve the efficiency in which Tyche simulations are performed, it is necessary to separate the core simulation components from the user interface. Once this is achieved, Tyche can be run from the command line by the user or any other type of automation script, such as a Windows batch file.

The ability to run the application from the command line would enable Tyche to be sent to processing applications which handle high-throughput computing environments for command-line applications. The application would also be able to respond to external commands sent to it by the user to start, pause, resume, abort or restart.

1.2 Outline

As a result of the development during the summer of 2007, Tyche has been split into a “suite” of three applications; this documentation is divided into three areas with two sections each. The areas for Tyche, the Simulation Editor and the Dashboard each contain a user’s guide and a programmer’s manual. In addition, the final section of this document contains feasibility studies on areas for future improvement.

The user’s guide for each section is intended to serve as a “transitional” user’s guide for those who are already familiar with previous versions of Tyche. The documentation will focus purely on changes to the interface and operation of the dialogs. The programmer’s manual sections aim to instruct future Tyche developers on the areas of development new to Tyche version 2.2, as well as provide inside information to advanced Tyche users. These sections can be skipped by casual users of Tyche.

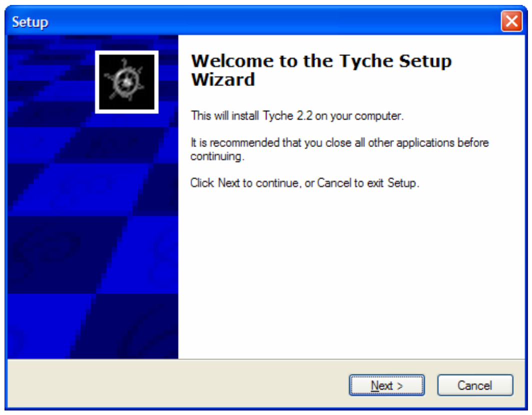
2. Installing Tyche Version 2.2

Tyche Version 2.2, like its predecessors, is a Microsoft™ Windows® based application which is designed to run on personal computers. The basic system requirements are Windows 2000 or later, a 1.5GHz processor or better and 386MB of RAM or more. When the user intends to run simulations locally with output data being saved to the hard drive, it is recommended to have at least 2GB of free disk space.

2.1 Installation Guide

Before the user starts installation, ensure the user's account has local administrative rights. To continue, launch the `tyche-v2.2-setup.exe` application and follow the prompts.

The first step will prompt the user to select the preferred language during the installation procedure. Once the preferred language has been selected, the user will be presented with the main welcome screen as illustrated in Figure 1.



Welcome to the Tyche Setup Wizard

This will install Tyche 2.2 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure 1. *Tyche Installer*

The user is advised to close all other applications before continuing. Closing other applications will ensure that they do not interrupt or negatively affect the installation sequence.

There are no settings or preferences to modify during the installation. Tyche will automatically be installed to C:\program files\Tyche 2.2. This default location will ensure that all registry changes for file type associations are correct.

Once installation is complete, the user will be asked to restart the computer. System file registration and file type association will be completed upon restart.

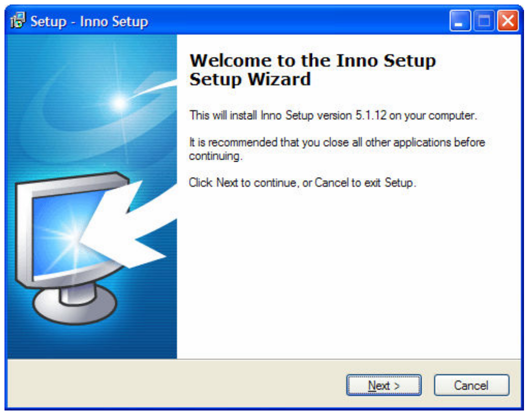
To start Tyche, click on the Tyche icon from the Tyche program group in the start menu.

2.2 Installation Programmer's Reference

Tyche is written in Microsoft™ Visual Basic® version 6.0. Visual Basic maintains projects of applications by clustering user interface and code modules within the same envelope. In previous versions of Tyche, only one Visual Basic project was involved. The Visual Basic Setup Compiler creates installation and setup files automatically for projects. However, Tyche version 2.2 is composed of three projects; one for each application within the Tyche version 2.2 suite. The Visual Basic Setup Compiler is not a viable option for creating the setup application for Tyche version 2.2 due to its multi-project nature.

Inno Setup [1] is a free and open-source setup application generator for Microsoft Windows programs. It creates installation and setup files for applications and produces an interactive setup wizard comparable to those generated by the Visual Basic Setup Compiler.

Download and install the Inno Setup Compiler to begin. When the user starts the Inno Setup Compiler installation the window illustrated in Figure 2 will appear.



Welcome to the Inno Setup Setup Wizard

This will install Inno Setup version 5.1.12 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure 2. *Inno Setup Installer*

Follow the prompts on-screen to install Inno Setup.

2.2.1 The Tyche Installation Script

The `tyche.iss` file is provided along with the application source files; this is the Inno Setup Script file. The iss file is text file that can be viewed with any text editor, but when opened with the Inno Setup Compiler, the file can be used to create an installation program using the Tyche application files.

There are six main sections within the Tyche installation script: Setup, Languages, Files, Icons, Run and UninstallRun. For basic information, see Table 1 for a full description of all the possible sections and parameters, consult the Inno Setup documentation which was included with the download of Inno Setup.

Table 1. *ISS File Section Descriptions*

SECTION OF ISS FILE	DESCRIPTION
[Setup]	This section contains global settings used by the installer and uninstaller.
[Languages]	This section defines the languages to make available to the Setup program. There are two isl files stored in the same directory as the iss file which define all the language constants for the setup application and may be modified by the developer to change any of the language phrasings. Note: This option will determine the language of the installer only – Tyche is only available in English.
[Files]	This section lists all files Setup is to install on the user's system.
[Icons]	This section defines any shortcuts Setup is to create in the Start Menu and/or other locations, such as the desktop.
[Run]	This section specifies any number of programs to execute after the program has been successfully installed. The Tyche 2.2 installer loads the file type association to the registry and will start the Dashboard during this section.
[UninstallRun]	This section specifies any number of programs to execute as the first step of uninstall process. The Tyche 2.2 installer executes a batch file which terminates all Tyche applications before continuing.

2.2.2 Compiling a New Setup File

In the [Setup] section of the iss, there is a parameter which appears as follows: `OutputDir=userdocs:Inno Setup Output`

This setting tells the Inno Setup Compiler where to save the resulting setup.exe file. In this case, the programmer's My Documents directory inside a subdirectory called `Inno Setup Output`.

Inside the Inno Setup Compiler, select Compile from the Build menu to start compiling the Tyche installer based on the specified parameters within the iss file.

3. User's Guide to the Tyche Editor

The functionality and operation of Tyche is fundamentally the same as in Tyche 2.0 and Tyche 2.1, so this guide is intended to be a transitional user's guide. Please see the documentation for Tyche 2.0 [3] and Tyche 2.1 [4] for additional information.

The user interface of Tyche 2.2 has been enhanced and upgraded to improve usability, flexibility and the overall end-user experience. What was previously referred to as simply "Tyche" is now referred to as the "Tyche Editor" because this application is now mostly concerned with the modification of input files and the viewing of output files.

3.1 Parent Window

The first display which appears upon running Tyche is the empty parent window illustrated in Figure 3.

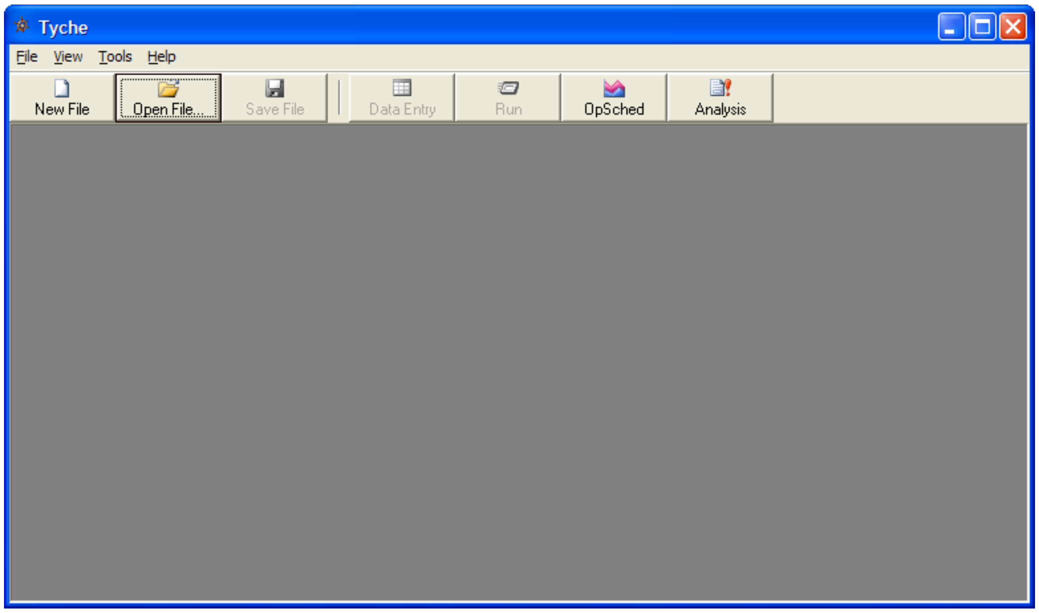


Figure 3. *Tyche Editor Parent Window*

The toolbar functions and menu commands provide access to all of the application's functionality. The toolbar is separated into two sections, one for file input/output and another for operational environments. The menus and environments are presented in the order in which they would normally be used. For complete menu descriptions, refer to Table 2.

Table 2. *Tyche Editor Menu Commands*

MENU		DESCRIPTION
File		The File menu contains file input/output access commands. Common shortcuts to these commands are replicated on the file I/O section of the toolbar.
View	Data Entry	This command is available only when an input file is open. The Data Entry Environment displays all objects in the currently open file. When the user opens a file, this environment window is displayed automatically.
	Run	This command is available only when an input file is open. The Run Environment opens the Tyche Simulation Editor window in order to start a new simulation for this data file.
	OpSched	The Operation Schedule Viewer allows users to graphically view the asset assignment choices made during a completed simulation run.
	Data Analysis	The Data Analysis environment allows the user to perform a statistical analysis on an output file generated from a simulation.
Tools		The Tools menu contains general options regarding Tyche preferences. Currently, it only contains the option to Confirm All Cancellations. By default, when the user makes changes to data entry dialogs, the user will be asked to confirm when cancel is pressed. When unchecked, this option will allow the user to cancel without being asked to confirm. There is room left for further options.
Help		The Help menu contains a reference to application documentation as well as version information.

3.2 Data Entry Environment

The Data Entry Environment window lists all the objects in the currently opened file including Capabilities, Asset Types, Fleets, Bases, Theatres, Scenarios and Scenario Types. For detailed information on all the aspects of the Data Entry Environment, consult Ref. [3].

All Data Entry windows opened through the main Data Entry Environment were previously set as children windows and all dialogs could be interacted with simultaneously. In Tyche version 2.0 and 2.1, all secondary editors must be completed and closed by the user before the user is permitted to continue.

The Data Entry Environment from previous versions of Tyche features a fixed window with non standard toolbars. The Data Entry Environment for Tyche 2.1 is illustrated in Figure 4.

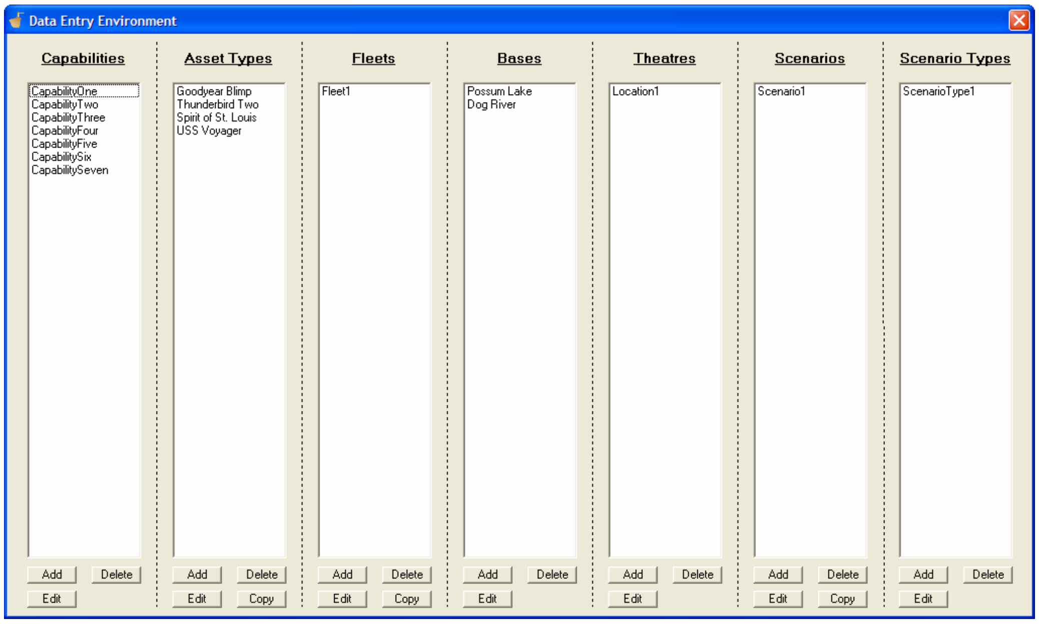


Figure 4. *Tyche 2.1 Data Entry Environment*

The Data Entry Environment for Tyche 2.2 is illustrated in Figure 5. The new Data Entry Environment features a new modern toolbar positioned at the top of the dialog and a resizable window which will increase or decrease the size of individual controls within respectively. In addition, the currently selected object type is highlighted in colour.

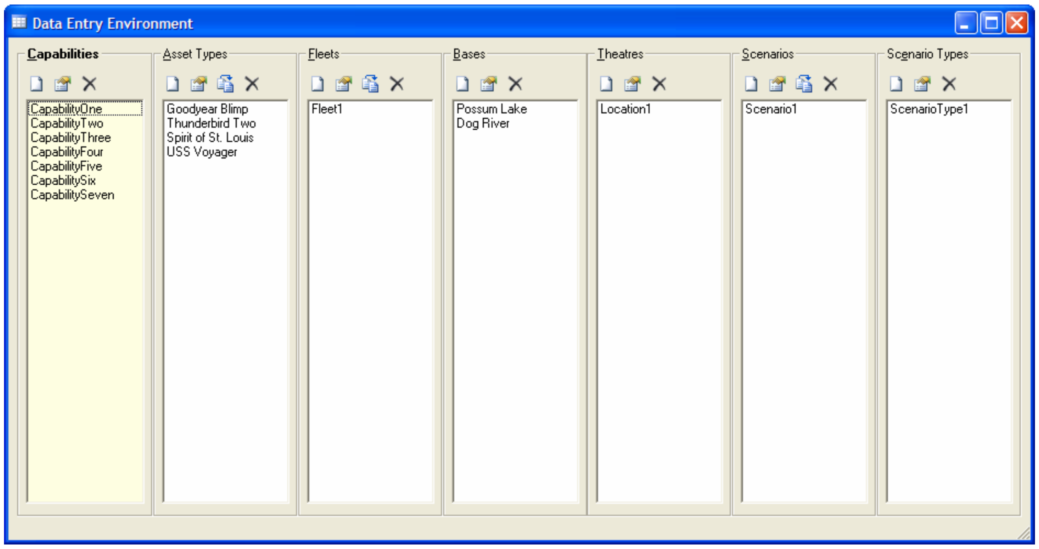
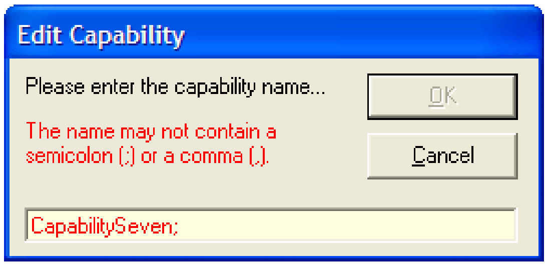


Figure 5. *Tyche 2.2 Data Entry Environment*

3.2.1 Capability

The new Capability dialog box features built in error-detection; see Figure 6. If the user inputs invalid data, the user will not be permitted to click the "OK" button. They will also be presented with an error message on screen.



Please enter the capability name...

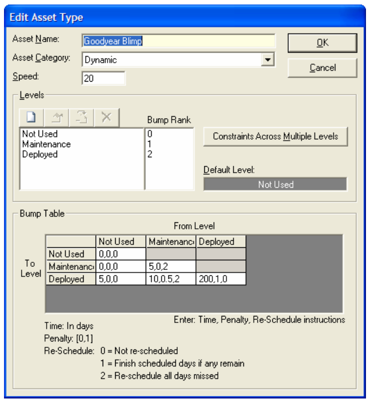
The name may not contain a semicolon (;) or a comma (,).

CapabilitySeven;

Figure 6. *Edit Capability Window*

3.2.2 Asset Types

The new Asset Type dialog box contains the same functionality as the previous version, but now appears more streamlined with more modern controls; see Figure 7. In addition, the dialog is slightly more compressed to make better use of space; this is mostly due to a toolbar replacing the series of normal large command buttons.



Asset Name: Goodyear Blimp

Asset Category: Dynamic

Speed: 20

Levels

Level	Bump Rank
Not Used	0
Maintenance	1
Deployed	2

Default Level: Not Used

Bump Table

From Level

To Level	Not Used	Maintenanc	Deployed
Not Used	0,0,0		
Maintenanc	0,0,0	5,0,2	
Deployed	5,0,0	10,0.5,2	200,1,0

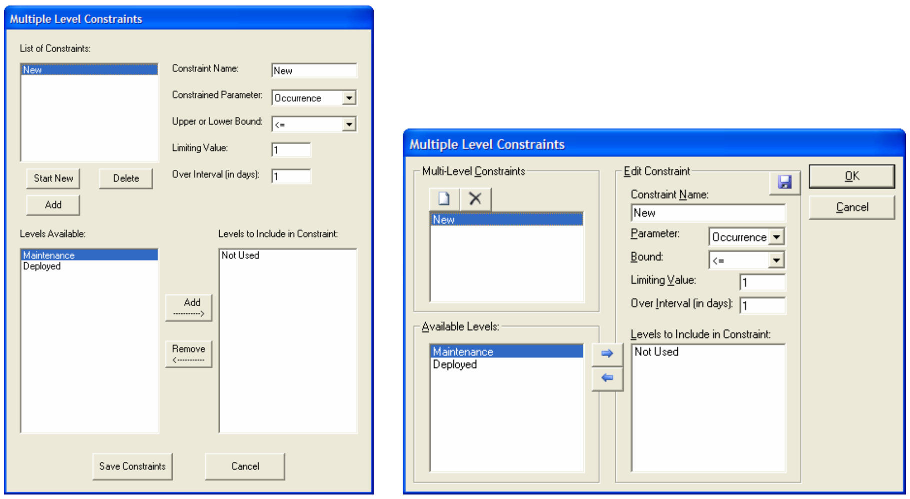
Enter: Time, Penalty, Re-Schedule instructions

Time: In days
Penalty: [0,1]
Re-Schedule: 0 = Not re-scheduled
1 = Finish scheduled days if any remain
2 = Re-schedule all days missed

Figure 7. *Edit Asset Type Window*

3.2.3 Multiple Level Constraints

The new multiple level constraints dialog contains a more intuitive design, as shown in Figure 8, and additional safety features. These safety features ensure that information changed by the user is not lost if the dialog's OK button is pressed before the user clicks the save button.



v2.1:

List of Constraints: New

Constraint Name: New
Constrained Parameter: Occurrence
Upper or Lower Bound: <=
Limiting Value: 1
Over Interval (in days): 1

Levels Available: Maintenance, Deployed

Levels to Include in Constraint: Not Used

v2.2:

Multi-Level Constraints: New

Constraint Name: New
Parameter: Occurrence
Bound: <=
Limiting Value: 1
Over Interval (in days): 1

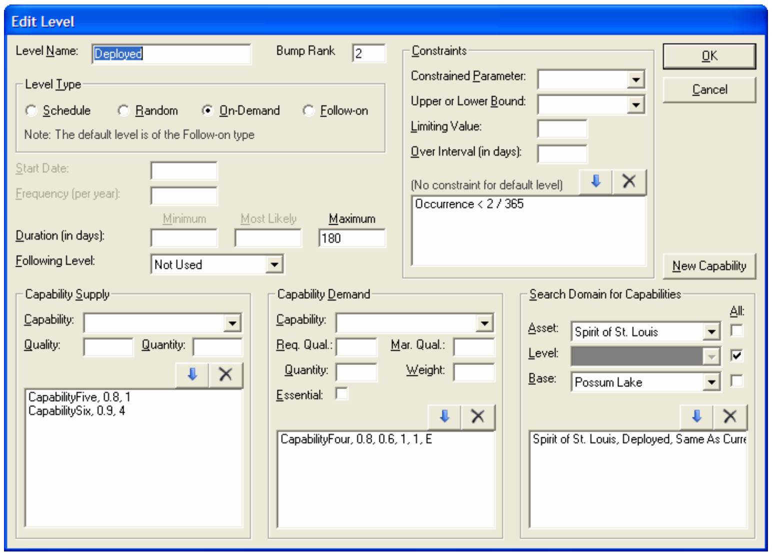
Available Levels: Maintenance, Deployed

Levels to Include in Constraint: Not Used

Figure 8. *Edit Multiple Level Constraints Windows (v2.1 left, v2.2 right)*

3.2.4 Levels

The Edit Level dialog box is essentially unchanged; see Figure 9. The addition of an "All" checkbox next to the Asset, Level and Base fields permits the user to specify all members of the specified field to be added to the Search Domain for Capabilities. This allows the user to save time when entering a large number of entries.



Level Name: Deployed
Bump Rank: 2

Level Type: ○ Schedule ○ Random [● On-Demand] ○ Follow-on

Note: The default level is of the Follow-on type

Duration (in days): Maximum 180

Following Level: Not Used

Constraints:
Constrained Parameter:
Upper or Lower Bound:
Limiting Value:
Over Interval (in days):
(No constraint for default level)
Occurrence < 2 / 365

Capability Supply:
Capability:
Quality:
Quantity:
CapabilityFive, 0.8, 1
CapabilitySix, 0.9, 4

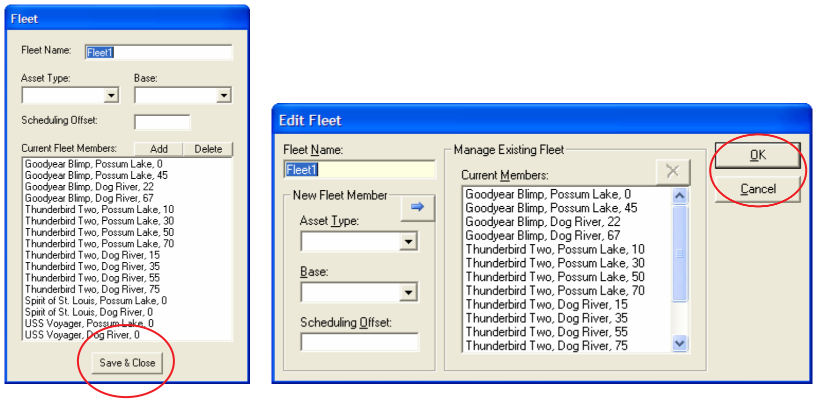
Capability Demand:
Capability:
Req. Qual.:
Mar. Qual.:
Quantity:
Weight:
Essential: ☐
CapabilityFour, 0.8, 0.6, 1, 1, E

Search Domain for Capabilities:
Asset: Spirit of St. Louis — All: ☐
Level: — All: ☑
Base: Possum Lake — All: ☐
Spirit of St. Louis, Deployed, Same As Curre

Figure 9. *Edit Level Window*

3.2.5 Fleets

Changes made on the Fleet Editor window are not binding until the user presses OK. In the previous version, there was no way to cancel the changes on the dialog once they have been made; see Figure 10. Compared to the previous version, the dialog now features a Cancel button to ensure that any changes are reversible.



Fleet Name: Fleet1

Current Fleet Members:

```
Goodyear Blimp, Possum Lake, 0
Goodyear Blimp, Possum Lake, 45
Goodyear Blimp, Dog River, 22
Goodyear Blimp, Dog River, 67
Thunderbird Two, Possum Lake, 10
Thunderbird Two, Possum Lake, 30
Thunderbird Two, Possum Lake, 50
Thunderbird Two, Possum Lake, 70
Thunderbird Two, Dog River, 15
Thunderbird Two, Dog River, 35
Thunderbird Two, Dog River, 55
Thunderbird Two, Dog River, 75
Spirit of St. Louis, Possum Lake, 0
Spirit of St. Louis, Dog River, 0
USS Voyager, Possum Lake, 0
USS Voyager, Dog River, 0
```

Fleet Name: Fleet1

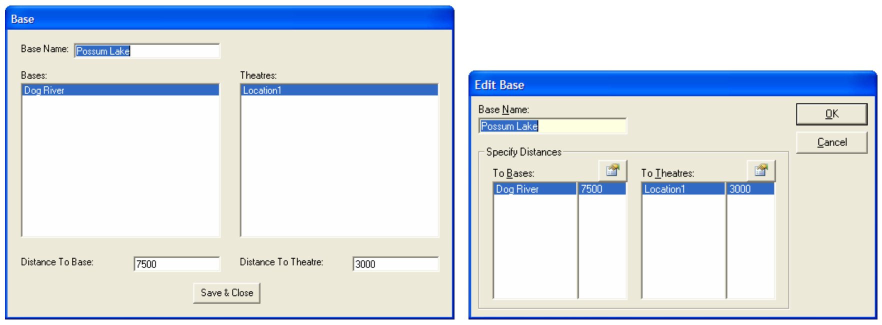
Current Members:

```
Goodyear Blimp, Possum Lake, 0
Goodyear Blimp, Possum Lake, 45
Goodyear Blimp, Dog River, 22
Goodyear Blimp, Dog River, 67
Thunderbird Two, Possum Lake, 10
Thunderbird Two, Possum Lake, 30
Thunderbird Two, Possum Lake, 50
Thunderbird Two, Possum Lake, 70
Thunderbird Two, Dog River, 15
Thunderbird Two, Dog River, 35
Thunderbird Two, Dog River, 55
Thunderbird Two, Dog River, 75
```

Figure 10. *Edit Fleet Windows (v2.1 left, v2.2 right)*

3.2.6 Bases

The Bases dialog box has also undergone significant changes; refer to both versions in Figure 11. All distances are now viewable concurrently at all times, whereas the previous version required the user to first select an entry before the value could be viewed or modified. A Cancel button has been added to ensure that the user can reverse changes made before OK is clicked. Finally, the window has been redesigned to be consistent with the other data editor dialogs.



Base Name: Possum Lake

Bases: Dog River

Theatres: Location1

Distance To Base: 7500

Distance To Theatre: 3000

Base Name: Possum Lake

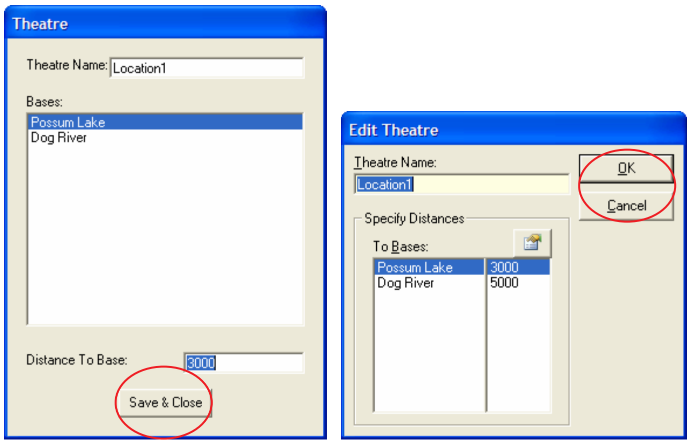
To Bases:	
Dog River	7500

To Theatres:	
Location1	3000

Figure 11. *Edit Base Windows (2.1 left, 2.2 right)*

3.2.7 Theatres

The Theatre editor has undergone changes identical to those of the Bases editor. It now displays all known distance information at all times and features a Cancel button, as shown in Figure 12. In addition, the dialog is slightly more compressed to make better use of space.


Theatre

Theatre Name: Location1

Bases:
Possum Lake
Dog River

Distance To Base: 3000

Edit Theatre

Theatre Name: Location1

Specify Distances

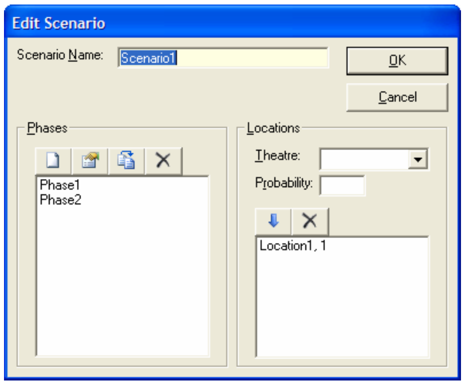
To Bases:

Possum Lake	3000
Dog River	5000

Figure 12. *Edit Theatre Windows (2.1 left, 2.2 right)*

3.2.8 Scenarios

The Scenario editor has undergone superficial changes only; see Figure 13. Large command buttons have been replaced with a toolbar to modernize the look of the dialog box.


Edit Scenario

Scenario Name: Scenario1

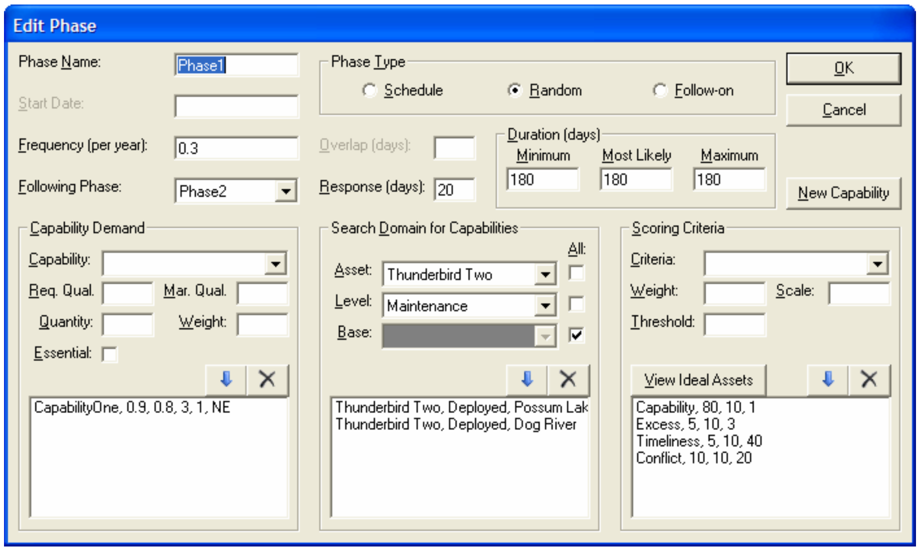
Phases
Phase1
Phase2

Locations
Theatre:
Probability:
Location1, 1

Figure 13. *Edit Scenario Window*

3.2.9 Phases

The Edit Phase dialog box is essentially unchanged; see Figure 14. The addition of an “All” checkbox next to the Asset, Level and Base fields permits the user to specify all members of the specified field to be added to the Search Domain for Capabilities. This allows the user to save time when there are a large number of entries.


Edit Phase

Phase Name: Phase1
Start Date:
Frequency (per year): 0.3
Following Phase: Phase2

Phase Type: Schedule [✓ Random] Follow-on
Overlap (days):
Response (days): 20
Duration (days): Minimum 180, Most Likely 180, Maximum 180

Capability Demand
Capability:
Req. Qual. / Mar. Qual.
Quantity: / Weight:
Essential:

```
CapabilityOne, 0.9, 0.8, 3, 1, NE
```

Search Domain for Capabilities
Asset: Thunderbird Two — All:
Level: Maintenance — All:
Base: — All: ✓

```
Thunderbird Two, Deployed, Possum Lak
Thunderbird Two, Deployed, Dog River
```

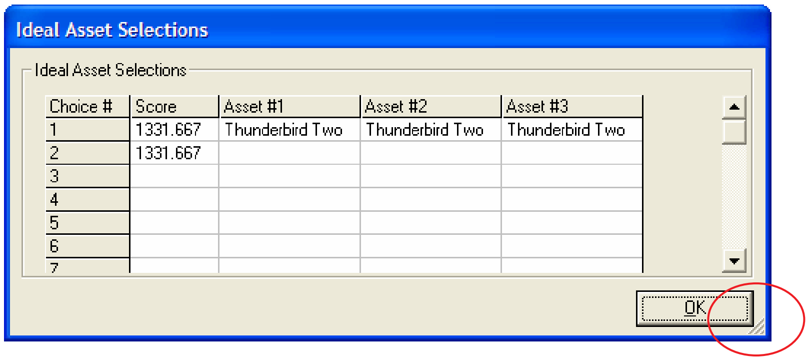
Scoring Criteria
Criteria:
Weight: / Scale:
Threshold:

```
Capability, 80, 10, 1
Excess, 5, 10, 3
Timeliness, 5, 10, 40
Conflict, 10, 10, 20
```

Figure 14. *Edit Phase Window*

3.2.10 Ideal Asset Selections

The Ideal Asset Selections window is re-designed to improve the ability of the user to read the text on screen; see Figure 15. Columns automatically expand to contain the text within the table and the dialog itself can be resized to display more of the table; as shown.


Ideal Asset Selections

Ideal Asset Selections

Choice #	Score	Asset #1	Asset #2	Asset #3
1	1331.667	Thunderbird Two	Thunderbird Two	Thunderbird Two
2	1331.667			
3				
4				
5				
6				
7				

Figure 15. *Ideal Asset Selections Window*

3.2.11 Scenario Types

A Cancel button has been added to the Scenario Types editor, shown in Figure 16, to ensure that the user can reverse changes once made before OK is clicked. Aside from this addition, the dialog has undergone only superficial changes to compress its space usage and to create a more consistent appearance.

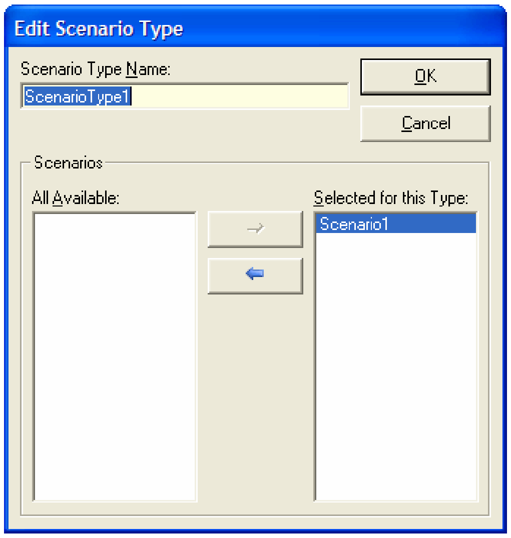


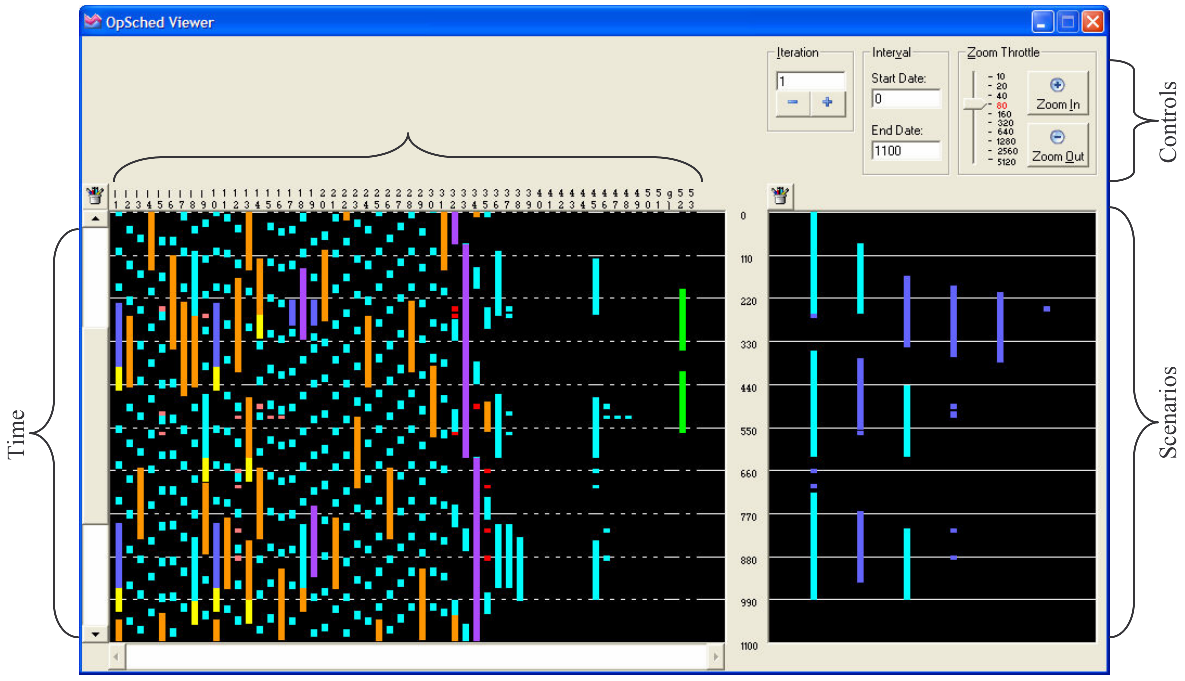
Figure 16. *Edit Scenario Window*

3.3 Run Environment

The Run Environment contains parameters for launching a new simulation based on the currently open input file. The Run Environment is now part of the new Tyche Simulation Editor. To setup simulation parameters and to launch runs, go to section 4.1.

3.4 Operation Schedule (OpSched Viewer) Environment

The OpSched Viewer, shown in Figure 17, allows users to graphically view the asset assignment choices made by Tyche after a simulation run is complete. The data is shown on a vertical chart that mimics a fleet operational schedule. For a comprehensive guide to the OpSched Viewer, refer to section 6.1 of Ref. [3] and section 5 of Ref. [5].



Iteration: 1

Start Date: 0

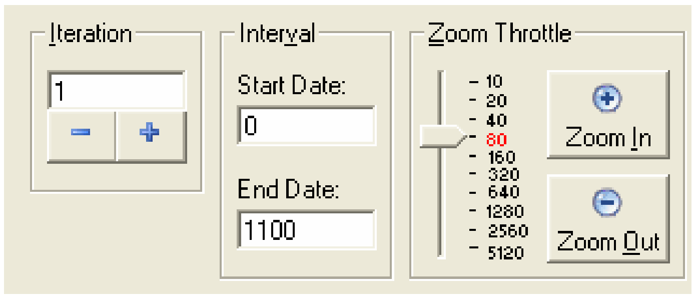
End Date: 1100

Figure 17. Operation Schedule (OpSched) Viewer

3.4.1 Consolidated Controls

From the end-user point of view, the less movement around a control the user is forced to perform, the easier it is to manipulate data on the dialog. Instead of all controls being spread out the window at literally all corners as in the previous version, the new version features a centralised control region.

The new region is pictured in Figure 18. It provides a more streamlined and intuitive user interface. Instead of requiring the user to press a command button after entering data into a text-entry field, the user is now permitted to simply hit the enter key to apply the changes.



Iteration: 1

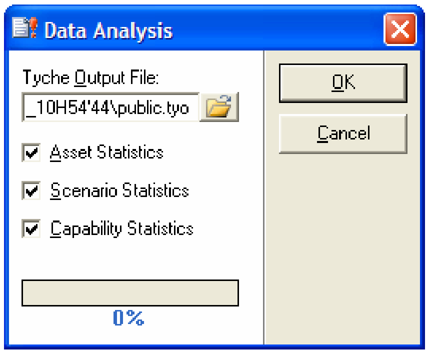
Start Date: 0

End Date: 1100

Figure 18. Consolidated OpSched Controls

3.5 Data Analysis Environment

The Data Analysis environment provides functionality to perform predefined statistical analyses on an output file generated by a simulation. This environment is pictured in Figure 19. The functionality of statistics generation is identical to previous versions of Tyche. For a complete and comprehensive guide to statistics generation, refer to section 7.1 of [3].



Tyche Output File: _10H54'44\public.tyo

☑ Asset Statistics

☑ Scenario Statistics

☑ Capability Statistics

Figure 19. *Data Analysis Dialog*

In previous versions of Tyche, the user had the option to perform statistics generation at the same time as setting up parameters for the simulation run itself. In version 2.2 of Tyche, the statistics generation is performed automatically after all simulations without prompting the user.

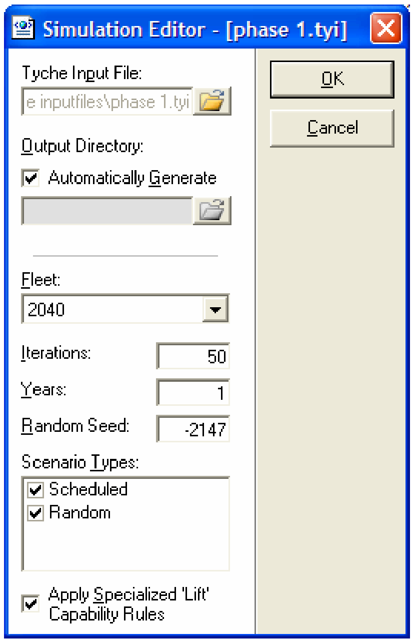
The Data Analysis Environment has not been removed from the Tyche main environment for legacy purposes, as well as to provide a way to manually generate new statistics without repeating the simulation run itself. The dialog appearance has been updated to make it consistent with the Tyche Simulation Editor window. Aside from this change, there are no further modifications to this environment.

4. User's Guide to Starting Simulations

For detailed descriptions of the events and activities which actually occur during a simulation, please see Section 5.1 of Ref. [3] and Section 2.2 of Ref. [4]. In Tyche version 2.2 the simulation execution task has been offloaded from the Tyche Editor and is handled jointly by the new Tyche Dashboard and the new Tyche Simulation Editor. In versions prior to 2.2, a user would press the Run button in the main Tyche window to run a simulation for the current file. This method continues to be supported, but there are additional ways to start simulations which will be discussed later.

4.1 The Tyche Simulation Editor

When the user opens the Run Environment from Tyche, the Tyche Simulation Editor pictured in Figure 20 will appear. Please note that a simulation cannot start if the input file does not contain any Fleets or Scenario Types; the user should check the input file for these attributes before continuing.



Tyche Input File: e inputfiles\phase 1.tyi
Output Directory:
[✓] Automatically Generate
Fleet: 2040
Iterations: 50
Years: 1
Random Seed: -2147
Scenario Types:
[✓] Scheduled
[✓] Random
[✓] Apply Specialized 'Lift' Capability Rules

Figure 20. Tyche Simulation Editor

4.1.1 Tyche Input File Field

When the Tyche Simulation Editor opens, the user will be prompted to select the input file from a location on disk. This field will contain the path to this file. Press the Browse (📂) button to select a different file.

Note: If the user selects an input file on a drive which appears not to be a locally mounted hard drive, the user will be presented with a warning which informs the user of the risk that the simulation may be interrupted or fail due to network instability.

4.1.2 Output Directory Field

By default, the location of the output directory is created in the directory containing the input file and named automatically using the following format:

`File [Fleet] (Iterations Years) Date Time`

For example, if the user opened a Tyche Input file named, `example.tyi` and started a simulation using the "`main`" fleet for one hundred iterations over five years, then the output directory will be created with the name:

`example [mainfleet] (100i 5y) 2007-07-03_14H50'51`

To manually specify the name and location of the output directory, deselect the Automatically Generate checkbox (☑) and press the Browse (📂) button; as shown in Figure 21.

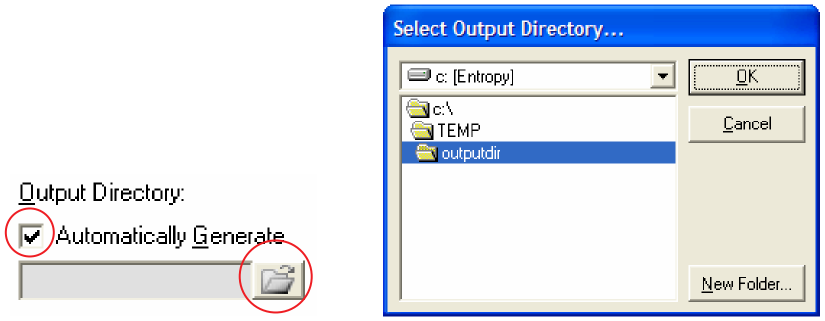


Figure 21. *Browse for Output Directory*

If the input file or output directory is on a drive which appears not to be a local mounted hard drive, the user will be presented with the warning that the simulation may be interrupted or fail due to network instability.

The output file, a copy of the input file, the file containing asset statistics, the file containing scenario statistics and the file containing capability statistics will all be saved to this new output directory.

4.1.3 Fleet Field

Select a Fleet from this drop-down box. By default, the first fleet listed in the input file will be selected.

4.1.4 Iterations Field

Specify the number of iterations simulation should run. The number of iterations must be a valid integer value greater than zero. By default, the iteration number will be extracted from the input file as saved by the last simulation to be completed using that value.

4.1.5 Years Field

Specify the number of years the simulation should execute during each iteration. The number of years must be a valid integer between one and eighty. By default, the year number will be extracted from the input file as saved by the last simulation to be completed using that value.

4.1.6 Random Seed Field

Specify the random seed the simulation should start execution from. The random seed must be a long value greater than -2,147,483,648 and less than zero. By default, the random seed is -2147.

4.1.7 Scenario Types List

Specify the scenario types to be utilized during the simulation. By default, all scenario types are selected. At least one scenario type must be selected.

4.1.8 Apply Specialized 'Lift' Capability Rules Checkbox

Selecting this checkbox will apply specialized rules for treatment of capabilities that contain “lift” at the beginning or end of their names. By default, this checkbox will be selected.

4.2 Starting the Simulation

Once the user clicks “OK” in the Tyche Simulation Editor window, the simulation will be initiated, the output directory will be created, a log file will be created in the output directory and the simulation task will be added to the queue. The parameters of the simulation cannot be modified at this point. See section 5 for information about managing the queue.

Each individual simulation runs as its own self-contained Windows task with a base priority of Low. This ensures that the user will experience no delay while using their computer for normal day-to-day tasks. Low priority simulations prevent a computer from being rendered unusable until all simulations are finished.

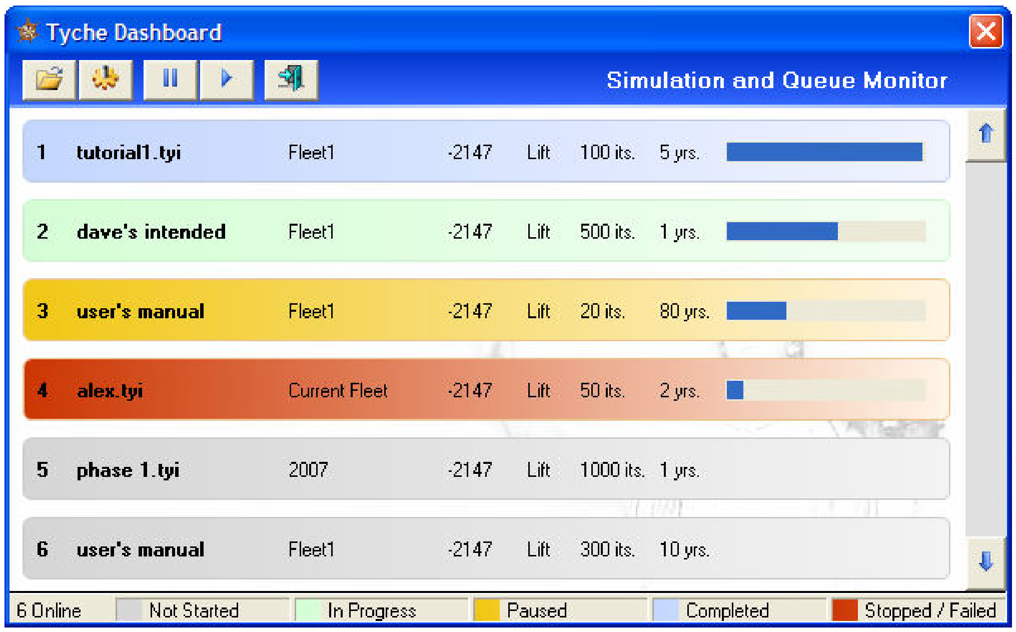
5. User’s Guide to Managing Simulations

A key feature of Tyche 2.2 is the new ability to stop, restart, pause and resume ongoing simulations. Prior to version 2.2 there was no way to perform simulation management of any kind.

5.1 The Tyche Dashboard

The Tyche Dashboard is the main user interface for interacting with all completed, running, paused, failed and queued simulations. The Dashboard manages the queue of simulations to ensure that only the maximum number of tasks is permitted to run concurrently. The Dashboard is pictured in Figure 22.

While Tyche simulations are able to continue unattended, the user can intervene by starting, pausing, resuming or aborting any particular task. The Dashboard also allows the user easy access to output directories, input files and log files. Finally, a simulation which has been aborted or failed can be resumed where it left off.



Tyche Dashboard

Simulation and Queue Monitor

1	tutorial1.tyi	Fleet1	-2147	Lift	100 its.	5 yrs.
2	dave's intended	Fleet1	-2147	Lift	500 its.	1 yrs.
3	user's manual	Fleet1	-2147	Lift	20 its.	80 yrs.
4	alex.tyi	Current Fleet	-2147	Lift	50 its.	2 yrs.
5	phase 1.tyi	2007	-2147	Lift	1000 its.	1 yrs.
6	user's manual	Fleet1	-2147	Lift	300 its.	10 yrs.

6 Online | ⬜ Not Started | 🟩 In Progress | 🟨 Paused | 🟦 Completed | 🟥 Stopped / Failed

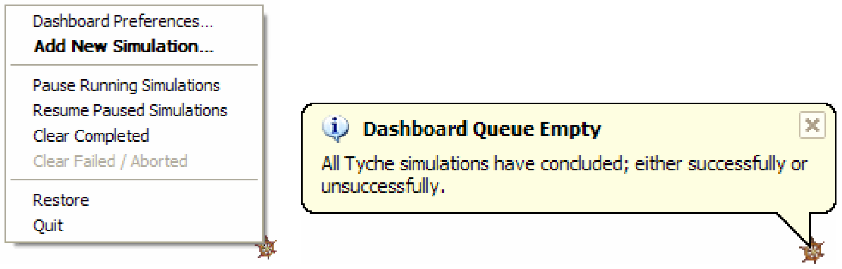
Figure 22. *The Tyche Dashboard*

Upon start-up, the Dashboard will display the list of the currently active Tyche simulations. The legend shows the colour a simulation will appear as, depending on its status. When the main window is closed by clicking on the “X” button on the top-right corner of the window, the Dashboard continues to run silently in background as an applet tray icon.

5.2 Applet Tray Icon and Toolbar

All of the Tool Bar functions are replicated in the context menu which is displayed when the user right-clicks the Applet Tray icon. This menu is also displayed when the user right-clicks on the background of the main window. See Figure 23 for illustrations of the Applet Tray icon functionality.

Open a new Tyche Input File to add to the queue.
Manage Tyche Dashboard Preferences.
Pause all Running Simulations.
Resume all Paused Simulations.
Quit Dashboard.



All Tyche simulations have concluded; either successfully or unsuccessfully.

Figure 23. *The Dashboard Applet Tray Icon with Menu and Balloon*

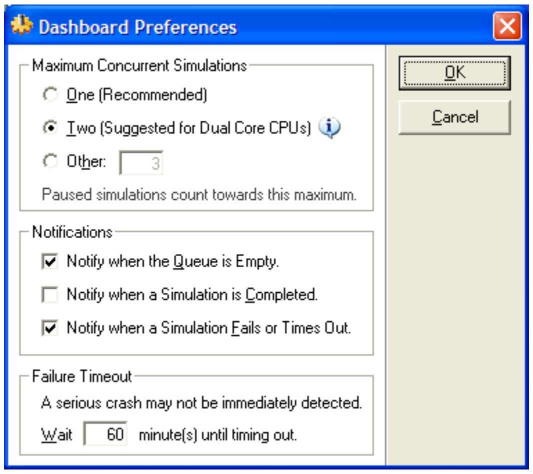
If the user has a large number of simulations in history, instead of manually clearing each entry, the user may select the option to clear all completed entries or to clear all stopped simulations.

The applet tray icon also serves as the source of balloon notifications regarding status changes of individual simulations. For example, if a simulation fails, the user may be notified as soon as the problem occurs.

When the user quits the Dashboard using the Quit ( ) button or by selecting the Quit command from the applet tray context menu, queued simulations will not be started but running simulations will continue uninterrupted. It is not recommended to exit the Dashboard while there are simulations are running or in queue, the user will not be notified if simulation errors occur.

5.3 Dashboard Preferences

The Dashboard's fundamental purpose is to ensure that the utilization of the system's processor computing time is optimal. The Dashboard preferences allow the user to manage the maximum number of concurrent tasks as well as the conditions under which to notify the user. The preferences dialog is illustrated in Figure 24.



Maximum Concurrent Simulations

( ) One (Recommended)

(•) Two (Suggested for Dual Core CPUs)

( ) Other: 3

Paused simulations count towards this maximum.

Notifications

☑ Notify when the Queue is Empty.

☐ Notify when a Simulation is Completed.

☑ Notify when a Simulation Fails or Times Out.

Failure Timeout

A serious crash may not be immediately detected.

Wait 60 minute(s) until timing out.

Figure 24. *Dashboard Preferences Dialog*

5.3.1 Maximum Concurrent Simulations

This option allows the user to manage the maximum number of concurrent tasks. If the computer does not contain a dual-core processor, the suggested maximum is simply one simulation at a time. Otherwise, two is the recommended maximum.

Click the information icon (ℹ) to open the System Properties preferences dialog box to display processor information for the workstation.

Additional simulations above these recommendations will increase the amount of time required to complete each running simulation. Following the recommended maximums will ensure that simulations complete in the minimum amount of time.

5.3.2 Notifications

When checked, the user will be presented with a Windows balloon notification when the specified event occurs. The possible notification options are as follows:

- **Queue is empty.**
  There are no simulations running or in queue. All simulations have either completed successfully or have failed. In either case, no computing power is being utilized. Checked by default.

- **Simulation is completed.**
  A single simulation has finished successfully. Unchecked by default.

- **Simulation fails or times out.**
  A single simulation has encountered a fatal error during processing, has not contacted the Dashboard during the timeout threshold or a new simulation could not be queued because its destination directory conflicts with an existing simulation. Checked by default.

5.3.3 Failure Timeout

If a simulation encounters a fatal error due to an error in the input file – such as a division by zero error or an overflow error – the crash will be reported to the dashboard and documented in the log file corresponding to that simulation. However, if the simulation instance is interrupted through some other unexpected problem, operating system failure or hardware failure, the Dashboard may not be able to immediately detect this failure.

Each simulation reports the current time to the Dashboard after each iteration. In order to ensure that simulations waiting in the queue are not standing by indefinitely, the timeout specifies the maximum tolerance before a simulation is deemed a failure and aborted. In these instances, the log file may not display any information regarding the failure.

The default timeout interval is thirty minutes. The timeout threshold must be set to a number of minutes greater than zero and less than one thousand. Note that a thousand minutes is roughly seventeen hours; if the simulation is taking longer than this threshold to complete a single iteration, there is probably a problem with the input file.

5.4 Administering Individual Simulations

Each simulation, depending on its current status, has a variety of commands available. The user may right-click a simulation entry in the Dashboard to bring up its available commands context menu; an example is shown in Figure 25.

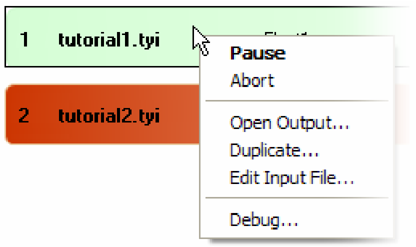
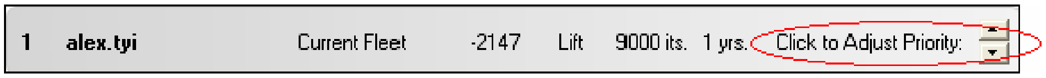


Figure 25. *Simulation Context Menu*

When issuing commands to running simulations, users may experience a lengthy response time. At the end of each iteration, a simulation checks for new commands from the Dashboard. The longer it takes for an iteration to complete – depending on factors of the input file – the longer it will take for a command to take effect.

For simulations that have not been started and are waiting in the queue, the user can adjust the order in which they are executed by adjusting their priority order; queued simulations at the top of the list are started first. Use the up and down arrows indicated in Figure 26 to either promote or demote a simulation's ranking. The control does not appear until the user selects the entry by clicking the item on the list.



1 alex.tyi Current Fleet -2147 Lift 9000 its. 1 yrs. Click to Adjust Priority:

Figure 26. *A Queued Simulation*

A Debugger is also available to view more detailed information about individual simulation instances. Refer to section 5.6 for a description of the Debugger.

In Table 3, a checkmark indicates that the command is available for a particular simulation's context menu. A solid box around a checkmark indicates the default action – in other words, the command which is executed when the entry is double-clicked.

Table 3. *Available Simulation Commands*

COMMANDS	**STATUS**				
	Completed	***In Progress***	***Paused***	***Stopped***	***Waiting***
Force Start					☑
Pause		☑			
Resume			☑	✓	
Abort		✓	✓		✓
Clear	✓			✓	
Open Output	☑	✓	✓	✓	
View Log	✓		✓	☑	
Duplicate	✓	✓	✓	✓	✓
Edit Input File	✓	✓	✓	✓	✓
Debug	✓	✓	✓	✓	✓

5.4.1 Force Start

The Force Start command overrides the maximum number of concurrent simulations permitted to begin the selected simulation immediately. This will not increase the normal maximum rule; when this forced simulation completes, a queued simulation will not take its place.

5.4.2 Pause

For a simulation that is running, the user may interrupt the processing while being able to resume at a later time. Pausing a simulation does not open a new slot for a queued simulation to begin, nor will the Dashboard ever automatically resume a paused simulation. The user must always manually resume a paused simulation for normal operations to continue.

5.4.3 Resume

For a simulation that is paused, the Resume command will return the simulation to normal processing. This is referred to as a "warm" resume.

For a simulation that has stopped, this command will attempt to recover the existing output data and restart the simulation from the last iteration successfully completed. This is referred to as a "cold" resume. A stopped entry can only be resumed if the original output file contains at least two iterations. This is due to the fact that the final iteration in any output file is discarded because it cannot be guaranteed to be correct and the recovery data for iteration is stored in the iteration before it. Therefore, an output file with only a single iteration recorded cannot be recovered.

5.4.4 Abort

The Abort command terminates a running or queued simulation. This will cause the Tyche instance running the simulation to cease execution immediately. The current iteration in progress (if running) will be written to file and the process will be stopped. The simulation can be restarted at a later time from the point where it left off using the resume command.

5.4.5 Clear

The Clear command removes the simulation from the Dashboard window when it has finished or stopped.

5.4.6 Open Output

The Open Output command displays the output directory in Windows Explorer to view the generated output files.

5.4.7 View Log

The View Log command opens the XML (e**x**tensible **m**ark-up **l**anguage) log file for the simulation using Windows Notepad to view simulation parameters with time stamped activity entries. This command is only available when a simulation is finished, paused or stopped; as viewing the log file is not recommended while there is a possibility it is being written to. A file access conflict could result in log data not being saved to the file. If the user wishes to view the log file while a simulation is running, the simulation must be paused first.

5.4.8 Duplicate

The Duplicate command starts a new simulation for the same Tyche Input File as the selected simulation. The simulation parameters will reflect the simulation parameters contained within the input file.

5.4.9 Edit Input File

The Edit Input File command opens the Tyche input file for the selected simulation in the Tyche editor.

5.4.10 Debug

The Debug command opens the Debugger for the selected simulation. The Debugger can assist in diagnosing and resolving simulation issues. See section 5.6 for a comprehensive guide to the Debugger.

5.5 Troubleshooting and Logging

All simulation parameters are saved to an XML file before the simulation begins. Tyche instances read this file to begin unattended simulations. This file is also used to store log entries written while the simulation is in progress.

Log entries appear at the bottom of the file in the order, in which they are written, dated with a time stamp, and appear like this:

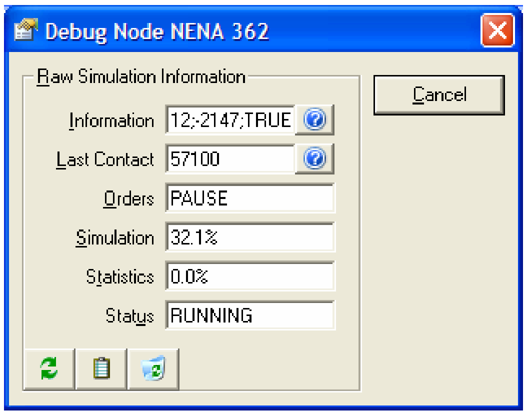
```
<log date="2007-08-07" time="15:42:33">Statistics
Generation Starting</log>
```

The XML log file can be used to attempt to diagnose problems with simulations. If the log file reveals that the simulation was instructed to abort and the user did not manually abort the simulation, the timeout threshold may need to be increased. The Dashboard sends an order to abort any simulation it has deemed a failure due to a timeout. See Annex A: XML Log Files for a comprehensive index of log entry descriptions.

5.6 The Debugger

While the Dashboard window presents a brief summary of simulation information, the Debugger presents all known information as reported by a selected simulation. The title of the Debugger window displays the unique code of four letters and three numbers which is used to identify the simulation instance.

The Debugger is a viewing utility only – no changes made to fields will have any affect on the simulation. See Figure 27 for an example of the Debugger window.



Information: 12;-2147;TRUE
Last Contact: 57100
Orders: PAUSE
Simulation: 32.1%
Statistics: 0.0%
Status: RUNNING

Figure 27. *The Debugger Window*

5.6.1 Refresh All Information Command

Click the Update button ( ) to update all fields on the Debugger to reflect the most up to date information as reported by the simulation.

5.6.2 Refresh and Copy to Clipboard Command

Click the Clipboard button ( ) to update all fields on the Debugger and copy all text fields to the clipboard.

5.6.3 Force Remove Command

Click the Remove button ( ) to force the removal of this entry. The option to Force Removal of the Simulation is present to enable the user to remove the simulation's entry from the Dashboard window if the user is confident that the simulation is defunct and no longer responds to commands.

For example, if a simulation is paused and a user kills the Tyche.exe manually with the Task Manager or reboots the computer, any orders to resume will not be processed and the clear command is not available for a paused simulation. Use this option as a last resort to perform housekeeping on the Dashboard window.

5.6.4 Information Field

The Information field presents a semicolon-delimited list of all simulation parameters. The user may click on the Information button (?) to view the information presented neatly for easy readability in a message box. Note that this is not a complete report of all simulation parameters, merely those reported by the simulation for the Dashboard to uniquely identify it.

5.6.5 Last Contact Field

The Last Contact field displays the number of seconds since midnight when the simulation last contacted the Dashboard. This information is used to determine if the simulation has timed out. A simulation has timed out when the difference between the last contact time and the current system time is greater than the user-defined timeout threshold. The user may click on the Information button (?) to compare the last reported time with the current time.

5.6.6 Orders Field

The Orders field displays the last command dispatched to the simulation by the Dashboard automatically (such as starting a queued simulation) or by a user manually.

5.6.7 Simulation Field

The Simulation field displays the percentage completion of the simulation itself. This component is weighted as 95% of the total progress being displayed on the main Dashboard window.

5.6.8 Statistics Field

The Statistics field displays the percentage completion of the statistics generation. This component is weighted as 5% of the total progress being displayed on the main Dashboard window. Statistics generation does not begin until the simulation itself is completed.

5.6.9 Status Field

The Status field displays the current mode of the simulation. This mode could be running, paused, killed, failed or waiting.

6. Tyche Programmer's Reference

Prior to version 2.2 of Tyche, there was only one Tyche application. As of version 2.2, three applications compose a "suite" which composes Tyche 2.2. Subsequent sections describe in detail the under-the-hood operations of all applications within the Tyche suite. The focus of this section is the offloading of simulation procedures from the main Tyche editor. A normal Tyche user, with no intention to perform any development work or advanced tasks on the Tyche project, will not need to read past this point.

6.1 Intercommunication and Interactions Overview

The first application within the Tyche "suite," the Tyche Editor, serves as the editor and viewer for Tyche input files and Tyche output files respectively. The second application, the XML Editor, creates script files containing simulation parameters and launches new simulations. The final application, the Tyche Dashboard, manages and monitors these simulations. This section describes, in general terms, how information is exchanged between the original application and the two new helper applications. See Figure 28 for the Tyche communication flowchart.

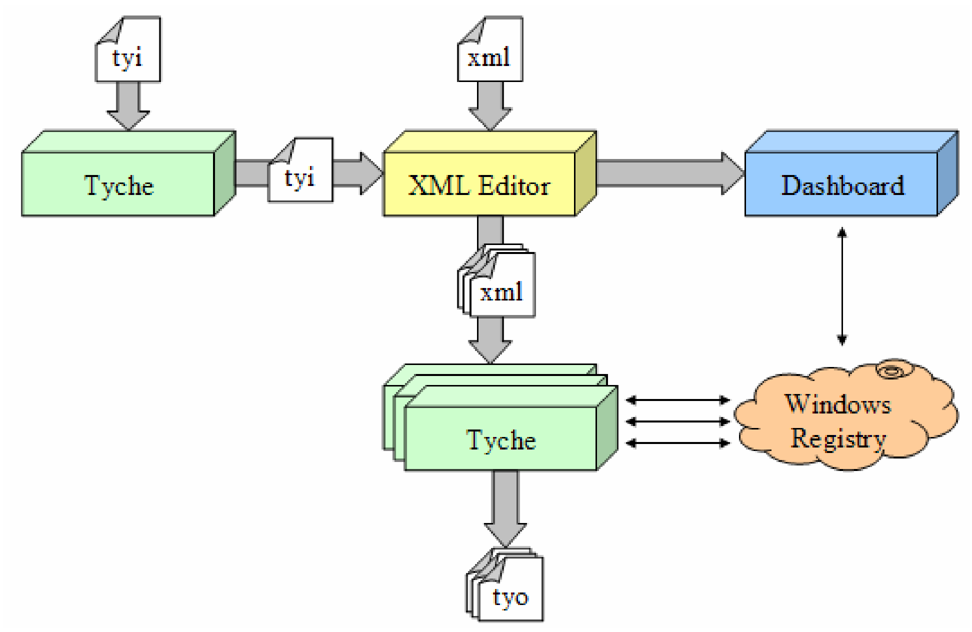


Figure 28. *Tyche Application Framework*

6.1.1 Tyche (tyche.exe)

Tyche continues to be the primary location users may open, modify and save input files. Tyche also contains the OpSched viewer which can be used to view the results contained within Output files.

As with previous versions of Tyche, a user can begin a simulation from the main user interface. The Tyche application continues to contain the procedures for running the actual simulation, but it no longer contains the code to permit the user to specify the parameters for the simulation.

When the user attempts to start a simulation from the Tyche application, the XML Editor application is launched using the location of the Tyche input file currently being edited as the argument on the command line.

When Tyche is launched using the location of an XML file as an argument on the command line, the silent automation execution procedure begins. During the simulation, Tyche saves and reads data in the Windows Registry to update status messages and process orders it has received.

6.1.2 XML Editor (XMLedit.exe)

The XML Editor is an enhanced standalone version of the former Run dialog in the previous version of Tyche. When launched with an argument of an existing XML file path, it displays the variables in the saved file to run the simulation specified. When launched with an argument to a tyi file, it loads all simulation parameters saved in the tyi file and defaults all remaining fields. If the XML Editor is launched without an argument, it will prompt the user to select either a tyi or XML file.

When the user clicks, “OK,” the XML Editor will update or create a new XML file and launch an instance of Tyche with an argument to the XML file. If the Dashboard is not already running, it will be started.

6.1.3 Dashboard (dashboard.exe)

Unlike Tyche and XML Editor, Dashboard does not accept arguments to files. The user can start a new simulation from within the Dashboard, but in actuality it is simply starting the XML Editor.

The Dashboard communicates with Tyche instances running in the background through the Windows registry.

6.2 Shared Code Modules

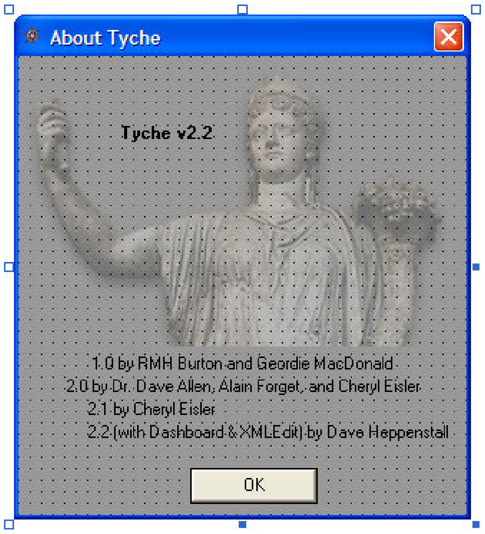
With the total of applications in the Tyche suite now up to three, there are undoubtedly areas of code which can be shared amongst multiple applications to reduce the total code volume and avoid large amounts of duplicate code.

Shared modules are stored in the `Shared Modules` directory within the master source code directory for Tyche 2.2 and set as read-only. It is critical that the utmost care be taken when modifying these files to ensure that all projects remain stable.

The two current shared modules are the about box and the simulation instance module.

6.2.1 About Box Form (frmAbout.frm)

With three applications, it is critical that the user be able to quickly verify that they are using the same version of each in order to avoid unpredictable version mismatch problems. Each application includes the same about box so that each new compile of any one application will display the correct version number intended after only making one change to the form shown in Figure 29.



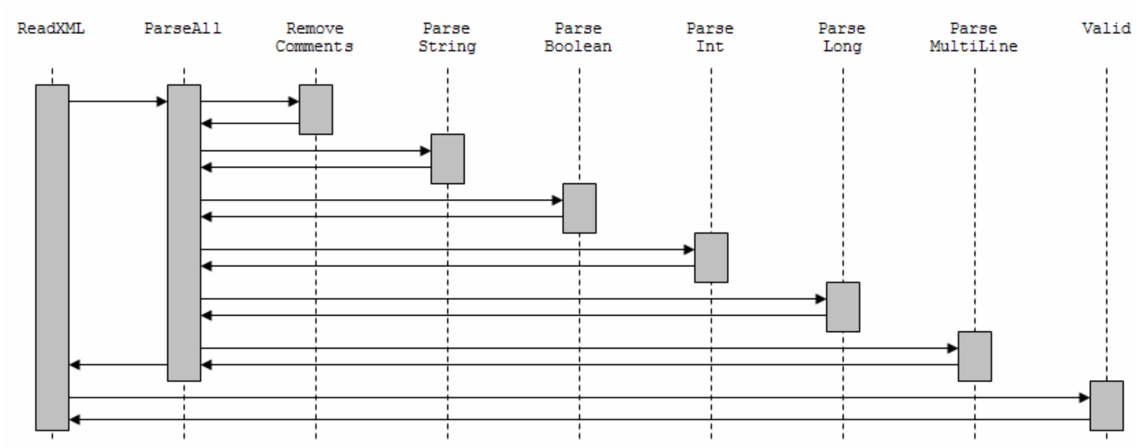
1.0 by RMH Burton and Geordie MacDonald
2.0 by Dr. Dave Allen, Alain Forget, and Cheryl Eisler
2.1 by Cheryl Eisler
2.2 (with Dashboard & XMLEdit) by Dave Heppenstall

Figure 29. *frmAbout.frm*

6.2.2 Simulation Instance Module (modSimulationInstance.bas)

This shared module is responsible for containing all relevant data and input parameters required to run a simulation which is later stored in an XML file to be read by Tyche when the simulation begins. This module contains all the necessary parsing, validation and output functions to handle safe input/output for XML files.

This module is used by the XML Editor to establish input parameters based on user data. The XML file is later passed on to Tyche once it is saved. When Tyche receives this XML file, it uses this module to parse the data therein and load the parameters for the simulation run. In both cases, the `ReadXML` method is called first. The following application timeline flowchart diagram indicates which methods of the module are called at what time; see Figure 30.



Method names across top; execution time read downwards

***Figure 30.** modSimulationInstance Timeline from ReadXML*

Note that the helper functions, `ParseString`, `ParseBoolean`, `ParseInt`, `ParseLong` and `ParseMultiLine` are called multiple times by `ParseAll` for each of the variable types, but are illustrated for simplicity in the figure above as one call each. If it is ever necessary to add a new parameter to the XML file, add a new public variable and a call to one of the helper parse functions to `ParseAll`. The `SaveAll` function dumps all input parameters directly into the local variables for the module and checks the new data for validity.

Refer to section 7 for comprehensive documentation on all parameters in the XML files used by Tyche.

6.3 Tyche Command Line Interface

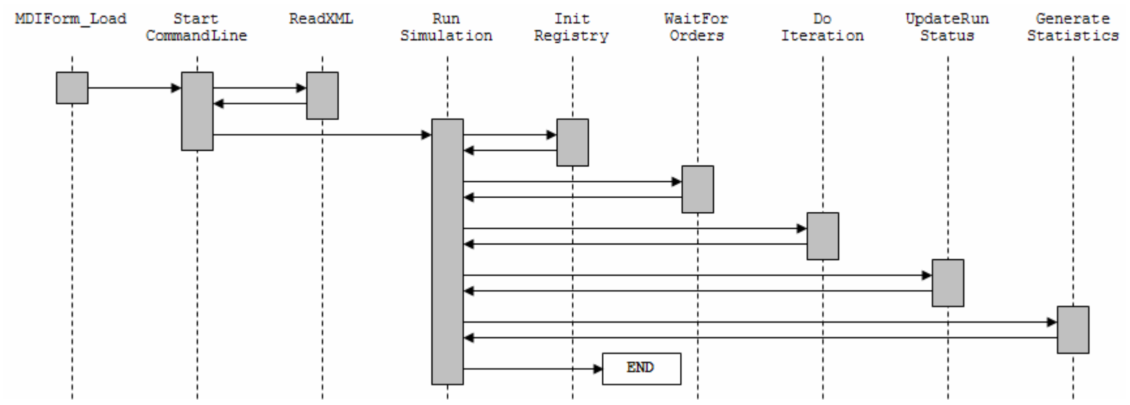
The traditional Run Environment has been removed from the user interface. The previous user interface form, `frmRun.form` is no longer present in the Tyche project. In version 2.2, when Tyche is launched the command line variable, command$, is checked to determine if the user has launched the application with a file path as an argument.

The expected and valid file types Tyche may receive are:

> Tyi – Tyche starts normally in the Data Entry Environment with the tyi file open.
>
> Tyo – Tyche starts normally in the OpSched Viewer with the tyo file open.
>
> XML – Tyche starts silently in the background in automation mode.

When an XML file is received on the command line, the data is parsed using the XML parser in the `Simulation Instance` module. The tyi file referenced within the XML file is then opened in the Tyche Data Entry Environment, but the graphical user interface does not appear on screen. Next, the simulation is initialized in the same way as the previous version of Tyche, but before the simulation begins, it waits for instructions from the Dashboard through the Windows Registry.

A new module, `Command Line` (modCommandLine.mod) has been written to handle the silent automation tasks in one location. See Figure 31 for the execution timeline.



Method names across top; execution time read downwards

Figure 31. *modCommandLine Execution Timeline from Tyche Startup*

Figure 32 illustrates the top-down flowchart for operations of Tyche while running in background mode.

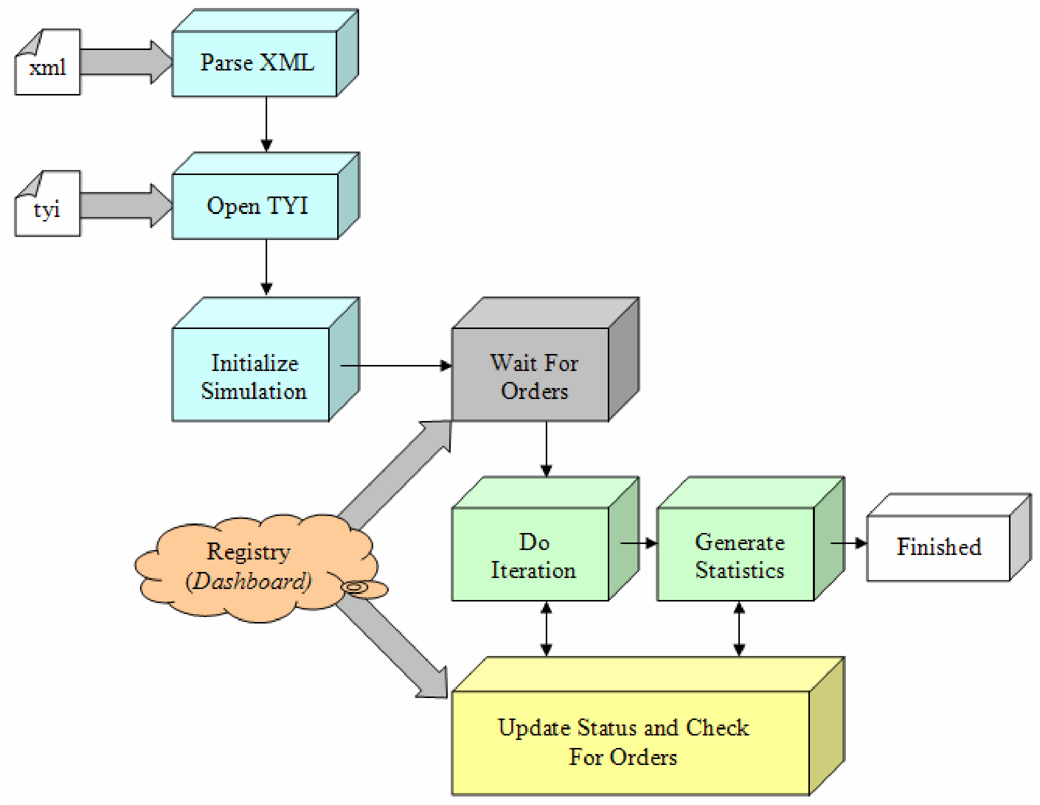


Figure 32. *modCommandLine Flowchart*

6.4 Automation Mode

As soon as Tyche starts up in automation mode, it generates a unique identification code which can be used to distinguish the background instance from all other background instances. See section 8.3 for information regarding where this – and other background information – is saved to the Registry for communication with the Dashboard.

7. XML Editor Programmer's Reference

To the end-user, the XML Editor does not appear to be a separate application; it appears to simply be a dialog box called "Simulation Editor," which is invoked from either the Dashboard application or the main Tyche application. The XML Editor is in fact a self-contained utility whose sole purpose is to create and modify valid XML files to store simulation parameters.

Figure 33 indicates how the XML Editor appears to the end-user as a simple dialog box.

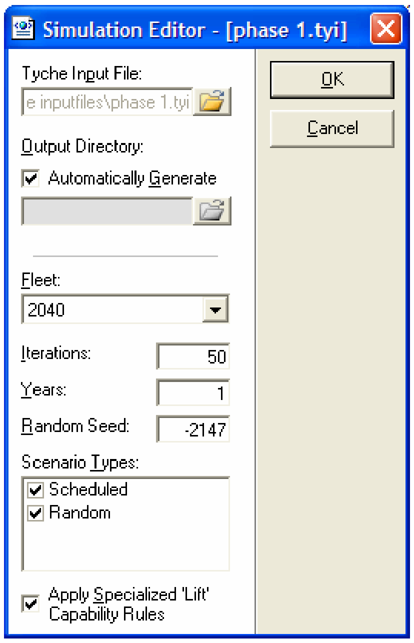


Figure 33. *XML Editor (Tyche Simulation Editor)*

7.1 Start-up

If the XML Editor is launched without a command line argument, the user will be prompted to browse for a proper input file. The vast majority of times, the XML Editor will be launched with an argument to a tyi file. This is how the main Tyche application and the Dashboard summon the XML Editor; by launching it with a path to a tyi file. The XML Editor will create a new XML file based on this tyi file when the user completes the simulation setup. Finally, if the XML Editor is launched with an argument to an XML file, the application will load all previously saved parameters and data from the Tyche input file to memory. When the user is finished, the changes will be saved to the existing XML file. These start-up operations described are illustrated in Figure 34.

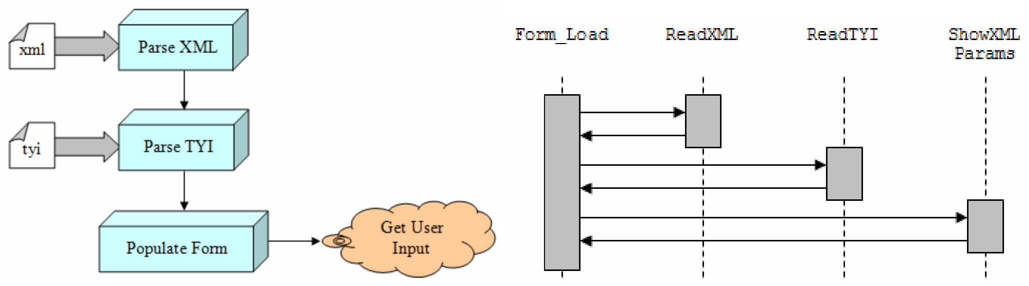


Figure 34. *Start-up Process for the XML Editor*

The XML Editor needs to read basic data from a tyi file in order to load information such as Fleets and Scenario Types housed therein. This is accomplished using the `ReadTYI` method. The `ReadXML` method is being called from the shared module, Simulation Instance. Further, it is in this shared module that all variables found in the XML file are stored.

The core operation of the XML Editor is packaged within the shared module, `Simulation Instance` (modSimulationInstance.mod). Refer back to section 6.2.2 for more detailed information on this module.

7.2 Job Submission

Once the user clicks the OK button on the XML Editor window, the application will perform an error check on the user input to ensure that there is no invalid data being saved to the simulation parameters. The fields being checked and their parameters for correctness are as follows:

- Exactly one **Fleet** must be selected.
- At least one **Scenario Type** must be selected.
- Number of **iterations** must be greater than zero and less than 32768.
- Number of **years** must be greater than zero and less than eighty one (81).
- **Random seed** must be a long integer greater than -2147483648 and less than zero.
- The **Output Directory** must be either automatically generated or manually selected.

The error detection occurs before the job is submitted. If an error occurs, the user is notified and they are prompted to correct the problem. Once the data passes basic error detection, the variables are saved to the Simulation Instance module, the XML file is saved to disk and the XML simulation task is submitted to Tyche. The job submission process is pictured in Figure 35.

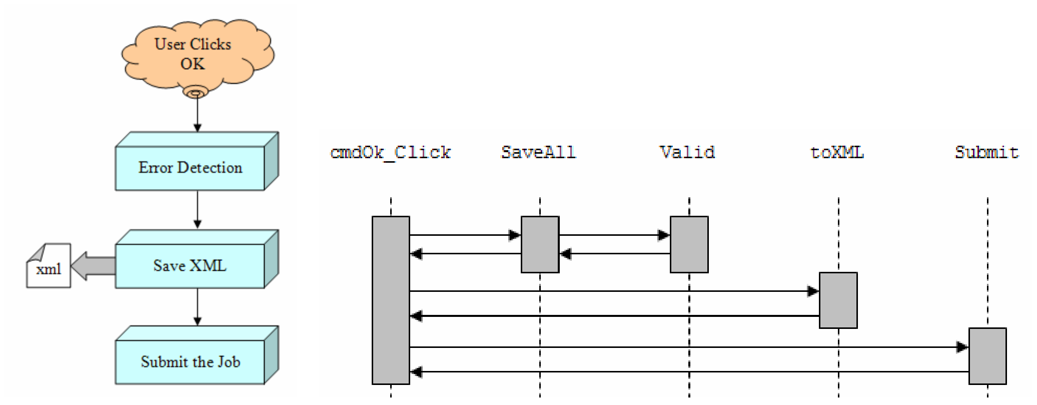


Figure 35. *Job Submission Process for the XML Editor*

The `toXML` method is another method contained within the Simulation Instance shared module. It returns the text as a String variable which is precisely the text which should be saved to the text file. See section 7.4 for a guide to XML statements used by Tyche.

Once the XML file is saved to disk, it can be submitted to the job queue. The `Submit` method is stored in a new file, the `Submit Job` module (modSubmitJob.mod), which contains built-in error handling and processes all directives required to submit the XML file to Tyche. Once a new Tyche instance has been launched with the path to the saved XML file as the argument, the XML Editor will launch a Visual Basic Script file, `Priority.vbs`, using the Windows Script Host application.

7.3 Adjusting Process Base Priority

The Visual Basic script file, `Priority.vbs`, is responsible for setting the priority of all running Tyche instances. The script cycles through the database of all running 32-bit processes, looks for any processes with the filename Tyche.exe and sets their base priority to “low.”

Specifically, this action is executed on the processes using the line, `Process.SetPriority (64)`, where 64 is the constant representing Low priority. See Table 4 for the index of all possible Windows priority levels.

Table 4. *Windows Base Priority Constants*

CONSTANT	PRIORITY DESCRIPTOR
64	Low
16384	Below Normal
32	Normal
32768	Above Normal
128	High

By requiring all Tyche instances to run at Low priority, the user will never experience any interruptions or slow responses while using their computer for day-to-day tasks. This is due to the fact that the majority of Windows applications run at Normal priority; therefore, when a process needs additional computing time, it borrows that time from the threads of lower priority processes. The trade-off is that simulations will take longer to complete depending on how actively the user makes use of their computer while the simulations run in the background. When the computer is idle, the simulations will use as much available computing time as the processor is capable of delivering.

7.4 XML Parameters

The XML Editor is the front-end utility for constructing XML files and launching Tyche using said files. However, there is nothing stopping a user from writing an XML file of their own and launching it using Tyche. One conceivable example would be if it is necessary for a user to run hundreds of simulations with slight variations in the simulation parameters. Instead of requiring the user to use the XML Editor, the user could write their own XML files using scripts, batch files, et cetera.

The `Simulation Instance` module performs comprehensive error detection to determine if there are any problems with the XML file which was not created by the automatic XML Editor generator. Syntax errors are highly unlikely when using the Editor and much more likely when written by a user. The XML files used by Tyche follow standard XML conventions and can be edited using any text editor. All automatically generated XML files contain comments used to describe each element present.

The file must begin with the line:`<?XML version="1.0" encoding="UTF-8" ?>`

The top level element must be: `<execution command="run">`

This element must be closed using the following at the end of the file: `</execution>`

Table 5 describes all possible parameters for an XML file to be considered valid by the Tyche XML parser. Also see Annex A: XML Log Files for a sample file.

Table 5. *Tyche XML Parameters*

PARAMETER NAME	MANDATORY	DESCRIPTION	PARAMETER CONSTANT
execution	✓	The top level element of the XML file. Contains all simulation parameters.	XML_TOPTAG
input-file	✓	The full path to the Tyche input file (tyi). String value; case insensitive.	TAG_INPUT_FILE
Fleet	✓	The fleet within the Tyche input file which is to be used. String value; case sensitive.	TAG_FLEET
Iterations	✓	Number of iterations to run the simulation. Integer value.	TAG_ITERATIONS
Years	✓	Number of years to schedule assets during each iteration. Integer value.	TAG_YEARS
Scenario-types	✓	A top-level element containing all scenario-type elements.	TAG_SCENARIO_TYPES
Scenario-type	✓	The name of a scenario type. Multiple elements of this tag are permitted. String value; case sensitive.	TAG_SCENARIO_TYPE
Seed		The Random Seed value. If omitted, the default value of -2147 is used. Long value; must be negative.	TAG_RANDOM_SEED
result-directory		The full path to an output directory. The directory itself does not necessarily need to exist, but the path to it must be valid. If omitted, the directory will be created within the same directory as the input file using an automatically generated name. String value; case insensitive.	TAG_OUTPUT_DIR
Generate		A top-level element containing apply-specialized-lift-capability-rules, asset-statistics, scenario-statistics and capability-statistics elements. May only be omitted if all elements contained are also omitted.	TAG_GENERATE
apply-specialized-lift-capability-rules		Contains "True" if the simulation should apply specialized 'lift' capabilities. Boolean.	TAG_APPLY_SPECIAL
asset-statistics		Contains "True" if the simulation should generate asset statistics. Boolean.	TAG_ASSET_STATISTICS
Scenario-statistics		Contains "True" if the simulation should generate scenario statistics. Boolean.	TAG_SCENARIO_STATISTICS

PARAMETER NAME	MANDATORY	DESCRIPTION	PARAMETER CONSTANT
Log		Contains human readable text. Properties of this element are "date" and "time" both of which also contain human readable date formats. Log entries typically describe important events relating to the simulation. These values are not read by Tyche and are intended for the user's use only.	TAG_LOG_ENTRY
start-point		It is not recommended that this value be manually entered into an XML file. This value is normally automatically detected and written during a cold resume procedure. If present, recovery-data must also be specified. The contained numeric value specifies the iteration number on which to start the simulation. Integer value.	TAG_START_POINT
Recovery-data		It is not recommended that this value be manually entered into an XML file. This value is normally automatically detected and written during a cold resume procedure. If present, start-point must also be specified. The contained value must be a recovery data list as read from the existing output file for the iteration defined in start-point. Array of Double values delimited by semicolons.	TAG_RECOVERY_DATA

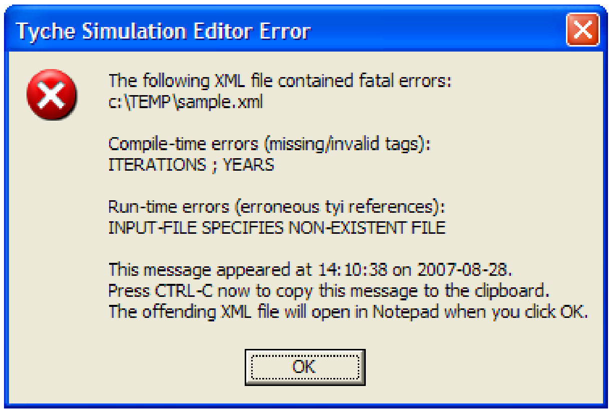
Once the programmer has written an XML file, the programmer can test it by opening the file using the XML Editor. The next section deals with validating XML files.

7.4.1 XML File Validation

To test the XML file, run the XML Editor application with the path to the XML file as a command line argument or start the application without an argument, select “Extensible Mark-up Language Files (*.XML)” from the File Type drop-down box and browse for the new XML file. If there are errors, the programmer will be presented with them in a message box and the XML Editor will quit.

Errors are grouped into two main subclasses: compile-time and run-time. Compile-time errors refer to parse errors on the XML file itself, whereas run-time errors refer to invalid data within the parameters. For example, if an input-file element is present but specifies a non-existent file, it would cause a run-time error.

Figure 36 displays the error in the instance that the mandatory elements, iterations and years, are missing and that the input-file element specifies a missing file.



The following XML file contained fatal errors:
c:\TEMP\sample.xml

Compile-time errors (missing/invalid tags):
ITERATIONS ; YEARS

Run-time errors (erroneous tyi references):
INPUT-FILE SPECIFIES NON-EXISTENT FILE

This message appeared at 14:10:38 on 2007-08-28.
Press CTRL-C now to copy this message to the clipboard.
The offending XML file will open in Notepad when you click OK.

Figure 36. *XML Read Error Message*

All possible compile-time and run-time errors are described in Table 6. All of the error detection and processing occurs within the `Valid` method within the `Simulation Instance` module.

Table 6. *XML Validation Error Messages*

ERROR STATEMENT	CAUSE
Compile-Time Errors	
INPUT-FILE	The input-file element is missing or contains no data.
FLEET	The fleet element is missing or contains no data.
ITERATIONS	The iteration element is missing, contains no data or is an invalid integer.
YEARS	The years element is missing, contains no data or is an invalid integer.
SCENARIO-TYPE	There are no scenario-type elements, or no scenario-type elements contain any data.
Run-Time Errors	
INPUT-FILE SPECIFIES NON-EXISTANT FILE	The input file specified does not exist or cannot be opened for reading.
INPUT-FILE SPECIFIES CORRUPT OR INVALID FILE	The input file is empty or does not contain text data.
FLEET SPECIFIES NON-EXISTENT FLEET	The fleet name specified in the fleet element does not exist in the Tyche input file. The fleet name element is case sensitive.
SCENARIO-TYPES SPECIFIES NON-ESISTENT SCENARIO TYPE(S)	At least one scenario-type named specifies a scenario-type that does not exist in the Tyche input file. The scenario-type element is case sensitive.

7.5 Cold Resume: Automated Simulation Recovery

For a simulation which has stopped, the Resume command sent by the Dashboard will attempt to recover the existing output data and restart the simulation from the last iteration successfully completed. What is actually happening is the Dashboard is launching the XML Editor using the path to the Tyche output file (tyo) as the command line argument. Due to the document naming convention, the path to the XML file can be intuited because it has the exact same name as the output file, but with a different extension. The user will be unaware that the XML Editor is involved at all in this process. See Figure 37 and Figure 38 for cold resume process diagrams.

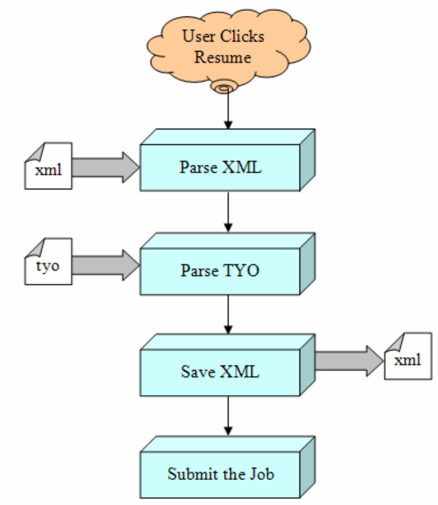


Figure 37. *Cold Resume Process Flowchart for the XML Editor*

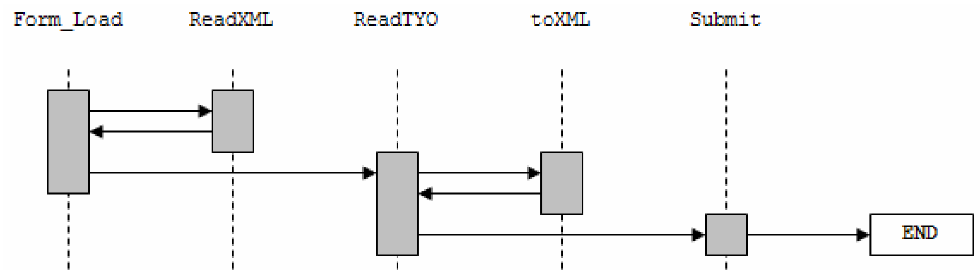


Figure 38. *Cold Resume Process Timeline for the XML Editor*

The `ReadTYO` method performs the following key operations:

- Removes the last recorded iteration from the output file. This is data removed because it may not be complete, and is considered unreliable.
- Reads the recovery data from the new “final” iteration in the output file. This recovery data contains all values of all variables used by the `Random` module (`modRandom.mod`) for the actual Tyche application. This recovery data is saved to the `recovery-data` element in the XML.
- The iteration which was considered unreliable is where the simulation must begin and is therefore saved to the `start-point` element in the XML.
- The XML file is overwritten with the new data, the path to the XML file is delivered to the `Submit` method which starts the simulation again.

Due to the aforementioned fact that the final recorded iteration in a file is considered unreliable, it is not possible to resume a simulation which has only a single recorded iteration. In this instance, no data can be considered reliable and there is nothing to recover. The Dashboard performs this check before the tyo file is sent to the XML Editor.

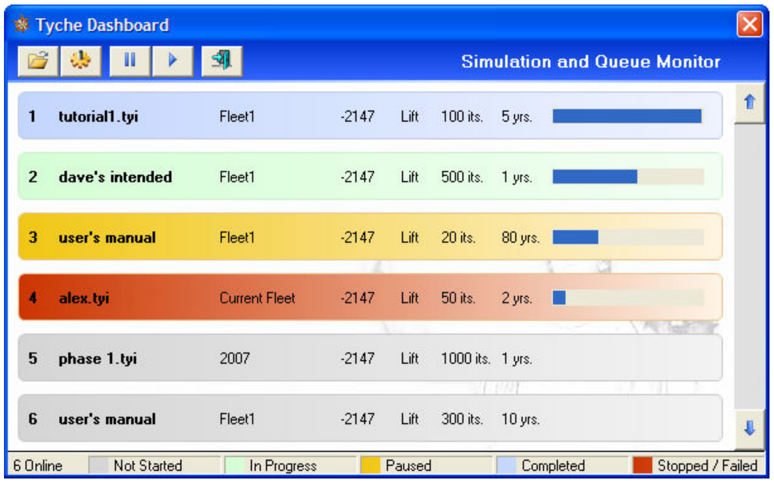
The output file of a simulation which has undergone one or multiple cold resumes is completely indistinguishable from an identical simulation which ran from beginning to end without interruption.

8. Dashboard Programmer's Reference

The Tyche Dashboard is the main user interface for interacting with all completed, running, paused, failed and queued simulations. The Dashboard manages the queue of simulations to ensure that only the maximum number of tasks is permitted to run concurrently.

No simulation can start unless the Dashboard is online and active to instruct it to begin. The Dashboard uses the Windows Registry to communicate with individual Tyche instances running simulations. When the Dashboard is started, it monitors individual Tyche simulations. The Tyche simulation environment data is stored in the registry. This monitoring is performed continuously over a predetermined interval.

The main Dashboard window displays a list of known simulation instances and appears in Figure 39.



1	tutorial1.tyi	Fleet1	-2147	Lift	100 its.	5 yrs.
2	dave's intended	Fleet1	-2147	Lift	500 its.	1 yrs.
3	user's manual	Fleet1	-2147	Lift	20 its.	80 yrs.
4	alex.tyi	Current Fleet	-2147	Lift	50 its.	2 yrs.
5	phase 1.tyi	2007	-2147	Lift	1000 its.	1 yrs.
6	user's manual	Fleet1	-2147	Lift	300 its.	10 yrs.

6 Online | ⬜ Not Started | 🟩 In Progress | 🟨 Paused | 🟦 Completed | 🟥 Stopped / Failed

Figure 39. Dashboard User Interface

The dashboard can be started with either of the following command line switches:

- "`dashboard.exe` **`-hide`**" instructs the Dashboard to start hidden in the background. The applet tray menu icon will appear but the main Dashboard window will not.

- "`dashboard.exe` **`-safe`**" starts Dashboard in safe mode. Safe mode is provided to allow the user access to the Dashboard if they are experiencing difficulty running the Dashboard. The Dashboard makes use of Windows DLL procedures to support balloon notifications and other features. If there are problems with these DLL files, the Dashboard may terminate unexpectedly in normal mode.

8.1 Monitoring Interval

The core monitoring procedure takes place in the `Timer` object. The interval is currently set to 1000 milliseconds (one second). This interval is chosen because it has a relatively light load on the processor but does not appear to be a long delay to the end user. During the monitoring interval, the Dashboard performs the following tasks:

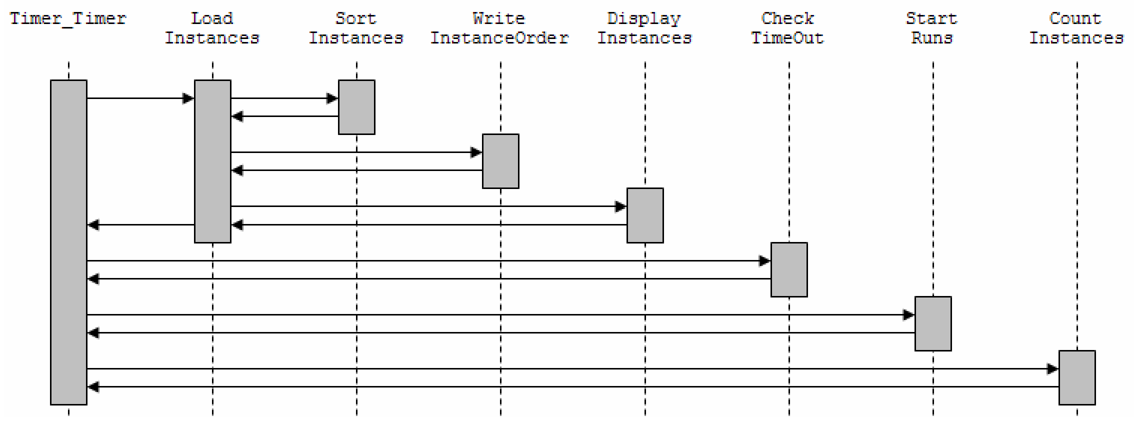
a. Loads information for all Tyche instances into local memory. These instances are stored in an array of `clsInstance` class modules and sorted by status. Finally, these instances are displayed on the graphical user interface;

b. Checks each running Tyche instance's last reported time. If the time it last reported in to the Registry is greater than the user-defined timeout threshold, then the simulation is assumed to have crashed, and an order is sent to the simulation to abort;

c. Determines if there are any available slots for instances currently in the queue and if so, starts the top-most waiting instance; and

d. Counts the number of simulations for each status type. These counters are compared to the last known counters so that the user is notified (if they have opted so in the Dashboard preferences) for all new completed simulations, new failed simulations or no queued/running simulations.

The execution flowchart illustrates the methods used in the order they are utilized for the monitoring interval; see Figure 40.



Method names across top; execution time read downwards

Figure 40. *Process Execution Timeline for the Timer Object (Monitoring Interval)*

8.2 User Preferences

The Dashboard stores general user preferences in the Windows Registry. This section describes exactly what settings in which locations of the Registry are recorded.

Visual Basic has pre-defined methods which exist to easily interface with the Registry to read and write application settings such as these. This being the case, settings are recorded on a per-user (not per machine) basis in the `HKEY_CURRENT_USER` top-level key. Specifically, `HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Tyche\Dashboard`. All values are stored in the Registry as String (`REG_SZ`) values.

Table 7 specifies all possible Registry settings used by the Dashboard. The Default column indicates what value is assumed if the setting is not present in the Registry.

Table 7. *Dashboard Registry Settings*

NAME	DEFAULT	EFFECT
Failure Timeout	"30"	An integer value between 1 and 999 inclusive which specifies the timeout threshold.
First Run	"TRUE"	If this is the first time the Dashboard has ever been run, display a notification to the user that the Dashboard has an applet tray icon.
First Simulation	"TRUE"	If this is the first time a simulation has started while using the Dashboard, notify the user that the Dashboard has launched it in the background.
Maximum Concurrent Simulations	"1"	The number of simulation "slots" permitted by the Dashboard.
Notify Upon Completion	"FALSE"	Notify the user upon each individual simulation completion using a balloon notification.
Notify Upon Failure	"TRUE"	Notify the user upon each individual simulation failure using a balloon notification.
Notify Upon Queue Empty	"TRUE"	Notify the user using a balloon notification when there are no running simulations and there are no queued simulations.

When the Preferences dialog is loaded, these settings are read from the Registry and are used to populate the form. Likewise, when the user is finished with the Preferences dialog, the settings are saved to the Registry. Settings such as "First Run" and "First Simulation" are automatically set to false after their respective notifications have occurred and are never reversed.

8.3 Simulation Instances

Individual Tyche instance data is stored within the Dashboard class module, `clsInstance`. This object reads the data contained within instance keys from `HKEY_CURRENT_USER\Software\VB and VBA Program`

`Settings\Tyche`. Within this `Tyche` key, there is a key named `Tyche Instances`. This key contains only one value, a String value which is also called `Tyche Instances`. This value contains a semicolon delimited list of all running Tyche instances using their unique identifiers – for example, "Node AAAA 000; Node BCBC 202; " with each instance name containing four randomly generated letters and three randomly generated numbers.

Each of these unique identifiers contains a key of its own within the Tyche key, such as `HKEY_CURRENT_USER\Software\VB and VBA Program Settings\Tyche\Node AAAA 000`. Within this sub-key, there are values for all relevant data regarding the simulation which are read by the Dashboard. An `Orders` value is written by the Dashboard and read by Tyche instances. This is the only value instances read during a simulation.

Section 5.6 describes the user's guide for the Debugger on the Tyche Dashboard. The Debugger examines all the values within this sub-key and presents them to the user. A comprehensive index of all variables within these node sub-keys are described in Table 8.

Table 8. *Tyche Instance Registry Settings*

NAME	CONTAINS	
Information	A semicolon delimited list of general simulation values. The list will read as the following: input-file; fleet; iterations; years; output-directory; random-seed; apply-lift-capabilities	
Last Contact	The number of seconds since midnight when the simulation last updated its status.	
Orders	The last command dispatched to the simulation by the Dashboard. Value will be one of the following:	
	NONE	Default order message when the instance has not yet been contacted by the Dashboard.
	WAIT	The simulation has not yet started, but it has been acknowledged by the Dashboard and placed into the queue.
	RUN	The Dashboard has moved the simulation from the queue into an available slot to begin processing or the user has resumed a paused simulation.
	PAUSE	The user has ordered the simulation to stand by for further instructions before proceeding.
	KILL	The Dashboard or the user has ordered this simulation to abort.
Progress: Simulation	A process indicator determining the number of completed simulation iterations divided by the number of total iterations.	

NAME	CONTAINS	
Status	The current status reported by the simulation. Value will be one of the following:	
	WAITING	The simulation has not yet started. May be started or aborted.
	RUNNING	The simulation is currently processing. May be paused or aborted.
	PAUSED	The simulation has started but is standing by. May be resumed or aborted.
	FINISHED	The simulation has concluded and is no longer running.
	KILLED	The simulation has been aborted and is no longer running.
	FAILED	The Tyche instance encountered an error and has crashed.

9. Future Development

During the development process for Tyche version 2.2, many areas of potential development were explored. Development tasks which were deemed infeasible to complete in the limited time allotted are described below.

9.1 Feasibility of High Throughput Computing

At present, individual Tyche simulations are only able to run off of one computer at a time. A typical computer has a great deal of idle computing time which, if properly utilized, could be shared among other computers. This section describes some possibilities for future development which could utilize additional computing resources.

9.1.1 Condor ®

Condor is a free and open-source software system which creates a High Throughput Computing environment by making use of available computing power of individual computers which communicate over a network. "The goal of the Condor® Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources." [1]

Originally designed for UNIX environments, Condor does support the Windows operating system. Ideally, a computing intensive application would be compiled with Condor's own libraries to allow the distributed execution to be more streamlined. Through the use of check-pointing and remote system calls, a job is be more reliable and makes better use of the network computing pool. Unfortunately, Tyche is programmed in Visual Basic, which is not supported by the native Condor libraries. This disadvantage requires that the job run in the "Vanilla" Condor universe as opposed to a "Standard" Condor universe. As a result, when a remote computer stops working on a job (for any number of reasons), the portion of that task may need to be discarded and restarted.

The Maritime Operational Research Team has several options for Condor:

- Deploy Condor on an existing Grid network.
- Deploy Condor on individual unclassified personal computers over the existing Wide Area Network (WAN).

[1] http://www.cs.wisc.edu/condor/

- Acquire additional computer hardware to create an isolated network on which Condor may be deployed.

9.1.1.1 Existing Grid Network

<Excised>

9.1.1.2 Personal Workstations on Existing WAN

<Excised>

9.1.1.3 Isolated Local Area Network

<Excised>

9.1.2 Enhanced Dashboard

Using Tyche version 2.2 a user is able to run simulations over a group of non-dedicated personal desktop workstations. The only issue becomes a legwork problem, as the analyst must manually visit all workstations in the pool which have Tyche installed to start simulations. To obtain progress and output data, the user must re-visit these workstations.

To simplify the process, it is possible that the Tyche Dashboard cold be made aware it is on a network and responsible for simulation requests from network users. While high-throughput computing applications will place continuous strain on the network, simulation delegation on an individual basis will place no more strain on the network than typical file sharing over mapped network drives.

Conceivably, a public file share could be created on the network and all Tyche applications on multiple workstations could be configured to be aware of that location. An analyst using a local Dashboard could start a simulation using the Tyche Simulation Editor, but enable a new checkbox named, “launch in remote queue.” This would cause the XML file not to be launched locally, but to be sent to the remote file share location. At a regular interval, these network-aware Dashboards would check if a Tyche XML file is in that directory. If the local Dashboard has computing power and a remote XML file has been posted, it will “take ownership” of that XML file. By moving the file to a local directory and launching the simulation normally on the local host, the posted job has been handled. Once the simulation is finished, the output data would be automatically copied back to the public file share on the network. The owner of the local workstation does not need to be aware that any simulation even took place. The local Dashboard would function normally for a remote user if they are interested in determining the status of computing demands placed on their workstation.

This conceptual plan requires a number of minor changes and enhancements to the Dashboard. To start, the Dashboard should be reconfigured to run as a Windows service which is always running; ensuring that all workstations with Tyche installed are always potential consumers of Tyche job requests from the network. If the owner of a workstation were to shut down their computer while simulations were running, the user would be notified that they would be interrupting other users' simulations. Upon condition that a simulation is stopped, it would be possible to upload the partially complete output data back to the shared network source.

There are additional considerations and unforeseen problems with this – or any – enhancement plan. For example, how would the output-directory value be handled if the simulation is running on a workstation with a different directory structure than the creator of the simulation? Until such a plan is implemented to create network-aware Dashboard applications, the user is still able to perform the legwork to accomplish the same end goal but with less efficiency and automation.

9.2 Functional Development Recommendations

A series of "wish list" items for Tyche did not make it into Tyche version 2.2. This section details the tasks which are recommended to be included within the development tasks for the next version of Tyche. Some recommendations are inherited from before Tyche 2.2 development began; see section 6 of Ref [4].

9.2.1 Streamlined Dashboard Queue

Instead of having the Tyche Simulation (XML) Editor automatically launch Tyche instances; it could submit the location of the simulation parameter XML file to the Windows registry. This would enable the Tyche Dashboard to read such a list and maintain a queue in this fashion. With a list of XML files on disk in the queue as opposed to actual running applications in the queue, this would reduce the number of running background tasks on the workstation as well as safeguard the queue if the operating system is restarted.

9.2.2 Phases and Ideal Asset Selections

The "View Ideal Assets" window allows the user to determine the effect on the scoring criteria on the assets selected for a scenario phase. There are a number of design changes and enhancements which this window should evolve towards:

- Filter results by allowing the user to select a Fleet as opposed to the current behaviour in which all assets from all fleets are included;

- Allow the user to view more than ten asset groups; and
- Upon right-click of a cell in the Ideal Asset Selections form, display the level number, base and list of matched capabilities for that particular asset during the particular scenario.

The Phases window also has potential areas for further development:

- Add a command to automatically populate the search domain based on assets that provide capability listed in the phase demand; and
- The phase following a previous phase should be dependant on the essential capability demands that were met during the previous phase. The following phase would not occur if the capability was not met in the previous phase. For more information on this recommendation, see section 6.2 of Ref. [4].

9.2.3 Error Detection Framework

In order to continue refining the end-user experience, a general error handling system should be constructed for the main Tyche application. The number of incidences in which Tyche crashes unexpectedly or behaves unpredictably needs to be reduced. This could involve the creation of a general Tyche log file for all errors.

9.2.4 Cosmetics

There is a growing collection of windows in Tyche which perform “smart resize” in which the contents within a dialog automatically expand to fit after a window is manually resized by the user. Add support for this feature to additional windows where appropriate.

In addition, scenarios in the Scenario window for the OpSched Viewer should be color coded (instead of scenario types).

9.2.5 OpSched Search

Development began on a search routine for the OpSched viewer. This routine would enable the user to search for assets or scenario types throughout the entire output file and graphically display the location of a matching asset or scenario.

Unfortunately, this development was not completed in time for its inclusion in Tyche 2.2. A non-functional front-end has been left with the project for development to resume at a later time, as shown in Figure 41.

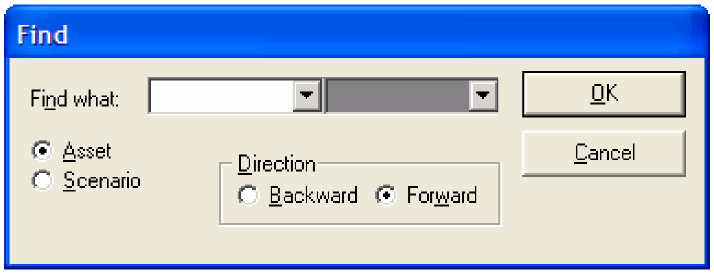


Figure 41. *OpSched Find Prototype*

9.2.6 Scheduling Offset

To improve the ability to generate data for scheduling offset values, two possible development paths are recommended:

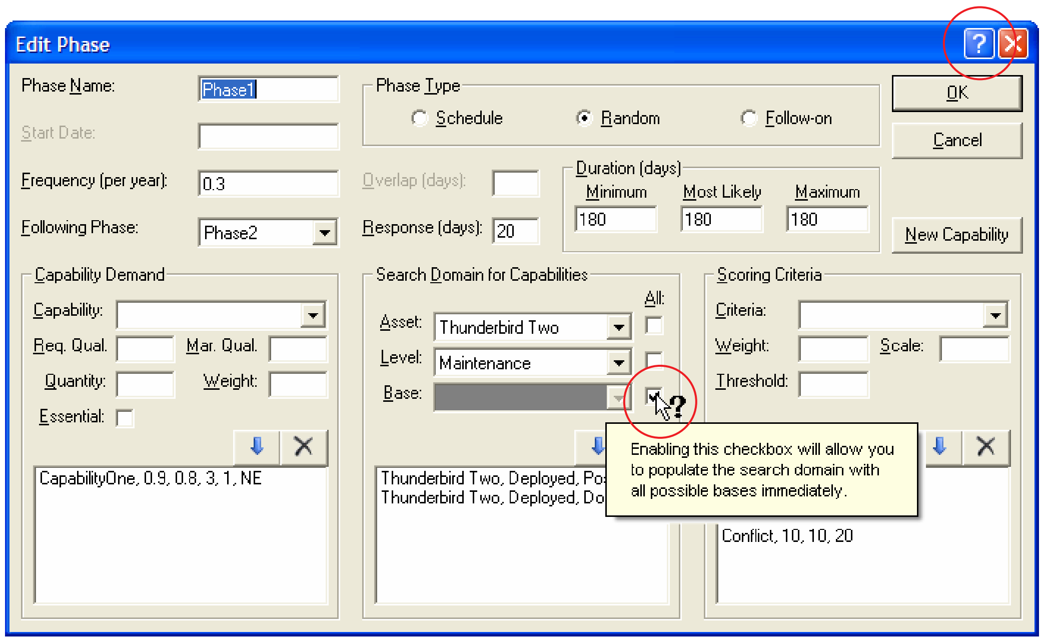
- Provide a routine for automatic generation pf scheduling offset for a given asset type using information from other member assets within the fleet; or
- Provide an interactive display (similar in theory to the OpSched viewer) for use before a simulation is run to display the level scheduling combined with scheduled scenarios. The visual representations would aid the user in determining overlap and modify the scheduling offset accordingly.

For more information on this recommendation, see section 6.2 of Ref. [4].

9.2.7 Context Help

In order for Tyche to become more friendly to new users, it must become faster and simpler for users to retrieve relevant context-based help while using the application. An example of where context sensitive help would be useful is shown in Figure 42.

The context help should also contain a terminology dictionary such that a user may rapidly learn the meaning of acronyms used in Tyche.



Phase Name: Phase1
Start Date:
Frequency (per year): 0.3
Following Phase: Phase2
Phase Type: Schedule [✓ Random] Follow-on
Overlap (days):
Response (days): 20
Duration (days): Minimum 180, Most Likely 180, Maximum 180

Capability Demand
Capability:
Req. Qual.: Mar. Qual.:
Quantity: Weight:
Essential:
CapabilityOne, 0.9, 0.8, 3, 1, NE

Search Domain for Capabilities
Asset: Thunderbird Two
Level: Maintenance
Base:
Thunderbird Two, Deployed, Po
Thunderbird Two, Deployed, Do

Enabling this checkbox will allow you to populate the search domain with all possible bases immediately.

Scoring Criteria
Criteria:
Weight: Scale:
Threshold:
Conflict, 10, 10, 20

Figure 42. *Context Help Prototype*

10. Concluding Remarks

The core operations of Tyche remain largely unchanged from the previous version. Simulation and statistics generation algorithms are identical to those of Tyche 2.1 and as a result, Tyche 2.2 is completely backwards compatible.

The new simulation operating procedure allows the normal and advanced Tyche user a great deal of flexibility. The user may write their own XML files, or write scripting programs to generate volumes of XML files to setup a large number of simulation tasks all at once.

Tyche simulations no longer operate as a runaway freight train – unstoppable until they reach their final destination. Now, Tyche simulations respond to real-time instructions from the user, report their progress at every step of the way in a log file which adheres to XML standards and can be resumed if it is interrupted; even if that interruption is due to a power outage. The disadvantage of providing real-time commands is that a simulation will take slightly longer to complete.

As a process though, running a large number of Tyche simulations is easier and faster than ever before. Tyche version 2.2 has already proven its utility to the Maritime Operational Research Team and further development is justified.

This page intentionally left blank.

11. References

1. Allen, D., Blakeney, D., Burton, R.M.H., Purcell, LCdr D. (2005). Fleet mix study: determining the capacity and capability of the future naval force structure. (Technical Report TR 2005-38). Defence R&D Canada – Centre for Operational Research and Analysis.

2. Russell, J. (1997-2007). Inno Setup (Online). http://www.jrsoftware.org/isinfo.php (1 July 2007).

3. Allen, D., Eisler, C., Forget, A. (2006). A user guide to Tyche version 2.0: providing a joint flavour to Tyche. (Technical Report TR 2006-14). Defence R&D Canada – Centre for Operational Research and Analysis.

4. Eisler, C. (2007). Improving the ability to model future maritime force structures: from Tyche version 1.0 to 2.1 (Technical Memorandum TM 2007-03). Defence R&D Canada – Centre for Operational Research and Analysis.

5. DeCorte, E. (2005). A programmer's guide to the user interface of Tyche version 1.0 – a maritime force structure model. (Contractor Report CR 2005-02). Defence R&D Canada – Centre for Operational Research and Analysis.

This page intentionally left blank.

Annex A: XML Log Files

Log Entry Descriptions

The following is a comprehensive index of all possible simulation log entries.

Table 9. *Annex 12.1 Log Entry Descriptions*

LOG ENTRY	DESCRIPTION
Saved simulation parameters to new XML file	The Tyche Simulation Editor has finished writing the simulation parameters to a new XML file. The time stamp on this log entry also indicates when the simulation was placed into the queue.
Simulation Starting: My unique ID is Node *AAAA 111*	Dashboard has started the queued simulation and the Tyche instance has generated a unique identification code which can be used to identify it.
Simulation Completed	The simulation has concluded and the Tyche output file is complete.
Statistics Generation Starting	The statistics generation procedure is starting. Any failures after this point will not affect the output file.
Built the assets used for collection and calculation	The assets data set has been built for statistics generation.
Built the phases used in collection and calculation	The phases data set has been built for statistics generation.
Collected data from output file	Data has been collected from the output file and statistics are ready to be calculated.
Completed Asset Statistics Generation	Asset Statistics have been saved to file.
Completed Scenario Statistics Generation	Scenario Statistics have been saved to file.
Completed Capability Statistics Generation	Capability Statistics have been saved to file.
Statistics Generation Completed	All statistics generation procedures have been completed successfully.
Run Finished	The simulation has concluded successfully and statistics generation has concluded successfully. There are no tasks remaining.

LOG ENTRY	DESCRIPTION
Received Order to Abort after Iteration *N*	The user manually instructed this simulation to abort and this order was confirmed by the simulation after the specified iteration was completed and written to the output file. OR The Dashboard has deemed this simulation a failure because it failed to update its status within the timeout threshold. Subsequently, it issued a “last ditch” effort to kill the simulation.
Received Order to Abort	The user manually instructed this simulation to abort while the simulation was paused. The iteration number is not displayed in this entry as it is the same iteration indicated in the pause order description.
Received Order to Resume	The user manually instructed this simulation to continue processing while it was paused.
Saved new starting position and other recovery data to XML file	The user has instructed a stopped simulation to resume and the Tyche Simulation Editor has saved additional parameters to the XML file to facilitate this request. The time stamp on this log entry also indicates when the simulation was placed into the queue.
Resuming Simulation from Iteration *N*: My unique ID is now Node *AAAA 111*	Dashboard has started the queued simulation and the Tyche instance has generated a new unique identification code which can be used to identify it. As the simulation was terminated, it is no longer has the same instance identification code.
== A FATAL ERROR HAS OCCURRED ==	A crash has been detected. This entry is always followed by additional log entry descriptors.
Fault thrown by object "*NNNN*" as reported by Err.Source	A crash was detected in a specific Visual Basic object as reported by the Err object’s Source property.
Error *N – Description*	A crash was caused by this specific error number with the reported description. These values are system constants used to handle trappable errors.
Failed During Statistics Generation	A crash was encountered during a specific stage in statistics generation.
Failed During Iteration *N*	A crash was encountered during an iteration.
The simulation itself did not fail; the .tyo is salvageable.	A crash was encountered after the simulation itself but during statistics generation. The output data is not corrupt and should be usable.
Run Failed	The Tyche simulation has crashed and did not complete successfully.

Sample XML File

The following is a sample XML file which can be used to automate simulations.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<execution command="run">
   <input-file>c:\temp\public.tyi</input-file>
   <fleet>2036</fleet>
   <iterations>50</iterations>
   <years>3</years>
   <seed>-38521</seed>
   <scenario-types>
      <scenario-type>PrimaryScenarioType</scenario-type>
      <scenario-type>SecondaryScenarioType</scenario-type>
   </scenario-types>
   <result-directory>c:\temp\public-output</result-directory>
   <generate>
      <apply-specialized-lift-capability-rules>True</apply-
specialized-lift-capability-rules>
      <asset-statistics>True</asset-statistics>
      <scenario-statistics>True</scenario-statistics>
      <capability-statistics>True</capability-statistics>
   </generate>
   <log date="2007-08-22" time="10:54:44">Saved simulation parameters
    to new XML file</log>
   <log date="2007-08-22" time="10:54:48">Simulation Starting: My
    unique ID is Node TSHN 483</log>
   <log date="2007-08-22" time="10:57:17">Simulation Completed</log>
   <log date="2007-08-22" time="10:57:17">Statistics Generation
    Starting</log>
   <log date="2007-08-22" time="10:57:24">Built the assets used for
    collection and calculation</log>
   <log date="2007-08-22" time="10:57:24">Built the phases used in
    collection and calculation</log>
   <log date="2007-08-22" time="10:57:25">Collected data from output
    file</log>
   <log date="2007-08-22" time="10:57:25">Completed Asset Statistics
    Generation</log>
   <log date="2007-08-22" time="10:57:25">Completed Scenario
    Statistics Generation</log>
   <log date="2007-08-22" time="10:57:25">Completed Capability
    Statistics Generation</log>
   <log date="2007-08-22" time="10:57:25">Statistics Generation
    Completed</log>
   <log date="2007-08-22" time="10:57:25">Run Finished</log>
</execution>
```

This page intentionally left blank.

List of symbols/abbreviations/acronyms/initialisms

HTC	High Throughput Computing
ISL	Inno Setup Language file
ISS	Inno Setup Script Installation File
OpSched	Operation Schedule
TYI	Tyche input file
TYO	Tyche output file
WAN	Wide Area Network
XML	Extensible Markup Language file