Overview

About This Guide .................................................................................................................xii
What's New with AppWorx 5.1 ...........................................................................................xiv

Section I: Introduction and Getting Started
1. Introduction to AppWorx ...................................................................................................3
2. Using the AppWorx Desktop ...........................................................................................21

Section II: Operations
3. Submitting Requests .......................................................................................................41
4. Viewing and Printing Output ...........................................................................................53
5. Monitoring and Managing Jobs ......................................................................................67
6. Viewing Summaries and Forecasts ..............................................................................119
7. Using the AppWorx Command Line Interface .............................................................139

Section III: Development
8. Creating Modules ...........................................................................................................157
9. Creating Chains .............................................................................................................189
10. Scheduling Modules and Chains .................................................................................225
11. Adding Prompts to Modules and Chains ...................................................................247
12. Assigning Predecessor Statements .............................................................................281
13. Working with Conditions .............................................................................................299
14. Defining Substitution Variables ..................................................................................345

Section IV: Administration
15. Defining Output Devices .............................................................................................361
16. AppWorx Security .......................................................................................................375
17. Working with Agents .....................................................................................................403
18. Administering Queues .................................................................................................421
19. Exporting and Importing Objects .................................................................................429

Appendices ........................................................................................................................449
Index .................................................................................................................................465
Overview
Contents

About This Guide....................................................................................................xii
What’s New with AppWorx 5.1 .............................................................................xiv

Section I: Introduction and Getting Started
1. Introduction to AppWorx .....................................................................................3
   1.1 A Modular Approach to Operations .................................................................4
   1.2 Overview of AppWorx Objects .........................................................................6
   1.3 How AppWorx Works .......................................................................................8
   1.4 Guidelines for Starting Out with AppWorx .....................................................10
   1.5 Starting Out with Existing Scripts .................................................................12
   1.6 AppWorx License Options ..............................................................................14
   1.7 Naming Convention Guidelines .....................................................................16
   1.8 Contacting AppWorx Technical Support .......................................................18

2. Using the AppWorx Desktop .............................................................................21
   2.1 Introduction to the AppWorx Desktop ..............................................................22
   2.2 Working in the AppWorx Windows .................................................................24
   2.3 Launching AppWorx .......................................................................................26
   2.4 Logging on to AppWorx ..................................................................................28
   2.5 Adding, Editing, andDeleting AppWorx Objects ...........................................30
   2.6 Viewing Object References ..........................................................................32
   2.7 Editing User Settings .....................................................................................34
   2.8 Linking to Online Manuals, AppWorx Support, and the About AppWorx Window .........................................................36

Section II: Operations
3. Submitting Requests ..........................................................................................41
   3.1 Introduction to Submitting Requests ..............................................................42
   3.2 Requesting Modules and Chains .................................................................44
   3.3 Editing and Submitting Modules and Chains ..............................................46
      3.3.1 Responding to List of Values (LOV) Prompts .........................................48
      3.3.2 Responding to Multiple Select (MS) Prompts ..........................................50

4. Viewing and Printing Output ............................................................................53
   4.1 Introduction to Viewing and Printing Output .................................................54
   4.2 Working with the Output Window .................................................................56
   4.3 Querying the Output Window for Specific Jobs ...........................................58
   4.4 Viewing Job Output with the File Viewer .....................................................60
4.5 Printing Job Output ................................................................................................................62
4.6 Using File Associations ............................................................................................................64

5. Monitoring and Managing Jobs .................................................................................................67
  5.1 Introduction to Monitoring and Managing Jobs ......................................................................68
  5.2 Using the Explorer Window ....................................................................................................70
  5.3 Using the Jobs Window ..........................................................................................................72
  5.4 Monitoring with the Status Bar and Object Icons .................................................................74
  5.5 Example: Finding an Aborted Job in a Chain .........................................................................76
  5.6 Viewing Jobs in the Backlog and History .............................................................................78
  5.7 Focusing the Backlog Display with Explorer .........................................................................80
  5.8 Changing Job Statuses in the Backlog ....................................................................................82
    5.8.1 Putting Jobs On Hold ....................................................................................................84
    5.8.2 Killing Jobs .................................................................................................................86
    5.8.3 Resubmitting Aborted and On Hold Jobs .....................................................................88
    5.8.4 Deleting Jobs ..............................................................................................................90
  5.9 Viewing and Editing Job Details ............................................................................................92
    5.9.1 Viewing and Editing Job Prompts ...............................................................................94
    5.9.2 Viewing Job Predecessors ...........................................................................................96
    5.9.3 Viewing Job Conditions .............................................................................................98
    5.9.4 Viewing Job Notes .....................................................................................................100
    5.9.5 Viewing Job Output ....................................................................................................102
  5.10 Working with Operator Logs ..................................................................................................104
  5.11 Monitoring Agents ..............................................................................................................106
  5.12 Managing Agents ...............................................................................................................108
  5.13 Monitoring and Managing Queues .....................................................................................110
  5.14 Querying for Jobs in the History .........................................................................................112
  5.15 Filtering the Backlog and History by Agents and Queues ..................................................114
  5.16 How AppWorx Handles System Failures ..........................................................................116

6. Viewing Summaries and Forecasts .............................................................................................119
  6.1 Introduction to Viewing Summaries and Forecasts ...............................................................120
  6.2 Viewing Chain Summaries ....................................................................................................122
  6.3 Using the Gantt Chart in Chain Summaries ...........................................................................124
  6.4 Working with a Chain Summary’s Flowchart ......................................................................126
  6.5 Printing Chain Summaries, Graphical Forecasts, and Chain Flows ......................................128
  6.6 Viewing a Forecast ..............................................................................................................130
  6.7 Viewing a Graphical Forecast ..............................................................................................132
  6.8 Setting the FORECAST Module Start Time(s) ..................................................................134
  6.9 Generating a Production Schedule ......................................................................................136

7. Using the AppWorx Command Line Interface ..........................................................................139
  7.1 Introduction to the AppWorx Command Line Interface .......................................................140
  7.2 Entering Generic Commands ...............................................................................................142
  7.3 Entering Backlog Commands ..............................................................................................146
  7.4 Entering Palm Commands ..................................................................................................150
7.5 Entering Other Commands

Section III: Development

8. Creating Modules

8.1 Introduction to Creating Modules
8.2 Defining Modules
8.3 Updating Unsaved Changes
8.4 Entering General Information for Modules
  8.4.1 Entering Program Information
  8.4.2 Defining Applications
  8.4.3 Defining Libraries
  8.4.4 Defining Program Types
  8.4.5 Commonly Used Program Types That Ship with AppWorx
  8.4.6 Entering Execution Options
8.5 Specifying Output Options for Modules
8.6 Setting Module Logins
8.7 Adding Notes
8.8 Assigning Roles to Modules
8.9 Copying Modules

9. Creating Chains

9.1 Introduction to Creating Chains
9.2 Defining Chains
9.3 Adding Chain Components
9.4 Updating Component Definitions
  9.4.1 Setting Component Schedules
  9.4.2 Setting Component Execution Options
  9.4.3 Specifying Component Output Devices
  9.4.4 Specifying Component Prompt Values
  9.4.5 Specifying Component Predecessor Statements
  9.4.6 Specifying Component Conditions
9.5 Viewing Chain Flows
9.6 Simulating Chain Flow Execution
9.7 Copying Chains
9.8 How AppWorx 5.1 Threads Chain Components
  9.8.1 Example: How AppWorx 5.1 Threads Chain Components
9.9 Why Use a Schedule Module?
9.10 Defining Schedule Modules

10. Scheduling Modules and Chains

10.1 Introduction to Scheduling Modules and Chains
10.2 Entering General Information for Schedules
10.3 Entering Schedule Frequencies
10.4 Entering Schedule Exceptions
## 10. Specifying Prompt Values for Schedules
- Specifying Prompt Values for Schedules ................................................. 234
- Defining Calendars .................................................................................. 236
- Example Schedules .................................................................................. 238
- How AppWorx Determines Eligibility ..................................................... 242

## 11. Adding Prompts to Modules and Chains
- Introduction to Adding Prompts to Modules and Chains ....................... 247
- What the Users See .................................................................................. 248
- Adding, Updating, and Deleting Prompts ............................................... 250
- Setting Minimum and Maximum Values for Prompts ............................ 252
- Defining Data Types ................................................................................ 254
  - Searching Your Database with SQL Statements ................................. 256
  - Using Referential Checking ................................................................. 258
- Adding a List of Values Prompt ............................................................... 260
- Adding a Multi Select Prompt ................................................................. 262
- How Multi Select Prompts Work ............................................................ 264
- Copying Prompts from Another Module or Chain ................................... 266
- Passing Values Through a Chain with Prompts .................................... 268
  - Example: Passing Values Through a Chain with Prompts .................. 270
  - Using Prompt Values in Chain Component Aliases ............................ 272

## 12. Assigning Predecessor Statements
- Introduction to Assigning Predecessor Statements ................................ 274
- Setting Virtual Workday Start Times ...................................................... 276
- Adding Predecessor Statements to Modules and Chains ....................... 278
- Grouping Predecessor Statements ......................................................... 280
- Deciding Between Predecessors and CHECK HISTORY Conditions ....... 282
- Example: Running a Series of Modules in a Virtual Day ....................... 284
- Example: Using Branching Logic to Determine Which Module Runs ...... 286
- Example: Running on the Successful Completion of a Chain Component ... 288

## 13. Working with Conditions
- Introduction to Working with Conditions ............................................. 289
- How AppWorx Processes Conditions .................................................... 291
- Adding, Editing, and Deleting Conditions ............................................. 293
- Selecting Condition Values ..................................................................... 295
  - Selecting Condition Timing ................................................................. 297
  - Selecting Condition Types ................................................................. 299
  - Selecting Condition Qualifiers ......................................................... 301
  - Selecting Condition Actions ............................................................. 303
  - Selecting Action Timing .................................................................... 305
  - Adding Substitution Variables and Replacement Values to Conditions 307
  - Using Replacement Values in Conditions ......................................... 309
- Using Conditions to Accomplish Common Tasks .................................... 311
  - Checking on Time Since Request ...................................................... 313
  - Checking if a File Exists ................................................................. 315
17.5 How Time Zone Shifts are Calculated .................................................................412
17.6 Defining Agent Groups and Assigning Agents to Them ..........................................414
17.7 Understanding Dynamic Load Balancing ...............................................................416
17.8 Managing System Records ....................................................................................418

18. Administering Queues ..........................................................................................421
18.1 Introduction to Administering Queues ....................................................................422
18.2 Defining Queues .....................................................................................................424
18.3 Defining Thread Schedules ....................................................................................426

19. Exporting and Importing Objects ...........................................................................429
19.1 Introduction to Exporting and Importing Objects ....................................................430
19.2 Export/Import Flowchart .......................................................................................432
19.3 Export/Import Terms ..............................................................................................434
19.4 Building Export Lists ............................................................................................436
19.5 Saving and Exporting Lists ...................................................................................438
19.6 Opening Imports and Mapping Objects ..................................................................440
19.7 Adding, Editing, and Deleting Maps File Entries ....................................................442
19.8 Example: Exporting a Module to Another AppWorx Instance ...............................444

Appendices ................................................................................................................449
Appendix A: Master/Agent Status Values .....................................................................450
Appendix B: Job Status Values .....................................................................................452
Appendix C: Troubleshooting Printer Problems ............................................................458
Appendix D: Regular Expression Tables ........................................................................460

Index .........................................................................................................................465
About This Guide

The AppWorx User Guide is a comprehensive procedures manual for AppWorx.

The AppWorx User Guide is written for operators, developers, and AppWorx system administrators. It is a comprehensive procedures manual that covers all aspects of AppWorx.

The User Guide is part of the AppWorx documentation set. Other manuals in the AppWorx documentation set include:

### AppWorx Document Set

<table>
<thead>
<tr>
<th>Manual:</th>
<th>Availability:</th>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation Guide</td>
<td>Print/PDF</td>
<td>Describes how to prepare your system for AppWorx and how to install AppWorx.</td>
</tr>
<tr>
<td>Technical Reference</td>
<td>Print/PDF</td>
<td>Describes tasks appropriate for a System or Database Administrator–focusing on module execution, run-time extensions, and basic scripting.</td>
</tr>
</tbody>
</table>

### Additional AppWorx Manuals

<table>
<thead>
<tr>
<th>Manual:</th>
<th>Availability:</th>
<th>Description:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppMaster User Guide</td>
<td>Print/PDF</td>
<td>Describes how to install and configure the AppMaster add-on product, create views for multiple masters and agents, and monitor jobs across masters, agents, and views.</td>
</tr>
<tr>
<td>AppWorx mySAP.com Extension Guide</td>
<td>PDF</td>
<td>Covers installation, setup and use of the AppWorx mySAP.com Extension</td>
</tr>
<tr>
<td>HP ITO Integration Guide</td>
<td>PDF</td>
<td>Describes how to install and configure the AppWorx HP ITO Integration product.</td>
</tr>
</tbody>
</table>

### Online Manuals


### PDFs on the Product CD and Support Site

PDF files for all AppWorx manuals are available on the AppWorx product CD, and the AppWorx Support Site: [http://support.appworx.com](http://support.appworx.com).
How This Guide Is Organized

The *AppWorx User Guide* is organized to approximate how most people learn to use AppWorx. First, it presents the basic skills first such as logging on to AppWorx. Covered next are the topics you use to operate AppWorx, such as requesting jobs, and viewing output. It then describes development topics such as how to add, update and delete objects (including modules and chains). Finally, the manual covers administrative topics, such as security and exporting and importing objects.

Unique Format

The manual is written as a series of topics, with most topics presented on two facing pages. Illustrations are always contained within the topic. These features make it easy to find where a topic starts and ends, and eliminates flipping pages to find an illustration.

1.1 Topic Heading

<table>
<thead>
<tr>
<th>Summary paragraph set off by horizontal rules.</th>
</tr>
</thead>
<tbody>
<tr>
<td>______</td>
</tr>
<tr>
<td>______</td>
</tr>
<tr>
<td>______</td>
</tr>
<tr>
<td>______</td>
</tr>
<tr>
<td>______</td>
</tr>
<tr>
<td>______</td>
</tr>
</tbody>
</table>

Each topic begins with a heading followed by a summary paragraph set off by horizontal rules. The summary paragraph states the key concepts presented in the topic. If a topic has a subtopic, the subtopic is also presented on two-facing pages. The topic heading is carried over to the subtopics, and is displayed in smaller letters above the subtopic heading.

In some rare cases a topic cannot fit onto two pages. To draw attention to these exceptions, we have included a continuation symbol...

In the lower right corner of the second page.

To get a quick overview of a chapter, read the summary paragraphs for each topic and look at the figures and figure captions.
What’s New with AppWorx 5.1

AppWorx 5.1 includes several new features and enhancements. Features and enhancements made to version 5.0.x are included at the end of the topic.

AppWorx Client Now Fully Java Enabled

AppWorx 5.1 includes a fully Java enabled client. The Java client does not need to be installed or updated, simplifying distribution and maintenance. With the 5.1 AppWorx client, you work from the AppWorx desktop to:

- Request modules and chains, track jobs, and view output.
- Monitor and manage the status of each agent and queue.
- Add, edit and delete objects with new administration selector windows. These windows are accessible as menu items on the AppWorx desktop, as icons on the toolbar, and as linking icons wherever that object type is assignable.
- Export and import AppWorx objects from one system to another.
- Access the AppMaster Administration window to add, edit, or delete AppMaster views.
- Set various user and session options.

New Interface Features

The Graphical User Interface (GUI) for the 5.1 AppWorx client has been greatly improved. The following features have been added to the AppWorx GUI.

- Tabs are used to “flatten” the interface, making it easier to navigate through various windows.
- A new toolbar is added to the desktop. The toolbar includes icons that link to the various AppWorx windows. It can be customized and/or hidden.
- A new taskbar is added to the desktop. The color of the Explorer/Jobs task button alerts you to the status of the AppWorx master, agents, and jobs running in the Backlog regardless of which window you are viewing. The taskbar can be hidden.
- New ToolTips provide a brief description of select AppWorx buttons, icons, and fields when you rest the mouse pointer over them. The ToolTips feature can be disabled from the AppWorx desktop.
- You can jump to a referenced object when adding or editing most object definitions.
- Assigned objects are automatically refreshed—no need to refresh lists.
- You can view all objects assigned to your user login with the “assigned objects” option.
• From the selector windows (which you use to add, edit, and delete objects), you can now:
  • Delete multiple objects.
  • See where objects are used.
  • Edit multiple objects at the same time. You can have two or more administration windows open at the same time in a client.
• You can print (and preview) tables, tree structures, and graphical views from the Explorer, Chain Summary, Graphical Forecast, and Chain Flow windows.
• You can view context-sensitive help that links you to the appropriate topic for each administration window.

New Graphical Analysis Package

An AppWorx 5.1 license with the Graphical Analysis Package ad-on option activated includes three new features:

• “Graphical Analysis Package” Chain Summaries
• Graphical Forecasts
• Chain Flows

The three features listed above are displayed in similar windows. Each window has three panes:

• An object tree, resembling the one in the Explorer window. It includes one or more chains and their components. You can expand and collapse the chains in the tree and select a job to bring it into focus in the table and graphical flowchart.
• A table of modules and chains selected in the tree or graphical flowchart. The table includes a Gantt chart used to show job execution statuses, run times, and estimated run times.
• A graphical flowchart of the chain(s) including any children. You can expand and collapse the chains in this view and select a job to bring it into focus in the table and the tree. You select display options including whether AppWorx shows predecessor relationships, execution order, and conditions.

“Graphical Analysis Package” Chain Summaries–Chain Summaries display the components belonging to a chain and any sub-chains, which are its children. They are accessed from the Explorer window.

Chain Summaries are included with all AppWorx keys-but are only fully featured with the Graphical Analysis Package.

The Gantt chart and graphical flowchart can be brought up for chains in the Backlog or History. You can right-click one or more modules/chains to edit their statuses or job details.
Graphical Forecasts—Graphical Forecasts display scheduled modules and chains. The data displayed in the Graphical Forecast window is generated and loaded into AppWorx by running the FORECAST module. When you create a schedule for the FORECAST module, you determine the timeframe of the forecast and how often it is run.

Chain Flows—Chain Flows allow you to simulate execution when building chains.

New Features and Improvements for Modules and Chains

AppWorx 5.1 includes many improvements to modules and chains. With AppWorx 5.1, you can:

• Add chains directly to other chains. In AppWorx 5.1, modules and chains that are added to a chain are referred to as “chain components”.
• Request chains directly from the Requests window. The need for schedule modules is now eliminated.
• Edit chain components as you add them to the chain.
• Use conditions defined for a module when adding the module to a chain.
• Optionally copy the conditions, notes, and/or schedules when copying a module or chain.
• Select a standard, OAE, SAP, or schedule template when defining modules to automatically assign default field values.
• Associate schedules with single modules as well as chains. For each module or chain you can:
  • Assign multiple named schedules.
  • Exclude a regularly scheduled running of a module or chain.
  • Choose whether to skip excluded jobs or run them on the previous or next day.
  • Add prompt overrides for values to a schedule.
• Add predecessor requirements to modules, chains, or chain components. Using predecessors, you can run the module or chain if a particular module or chain has, succeeded, failed, or completed since the last “virtual workday” (a start time set by an AppWorx module).
• Enter an average run time for a module, chain, or chain component. Average run times are displayed in the Gantt chart for Chain Summaries, Graphical Forecasts, and Chain Flows.

Changes to Other Objects

AppWorx 5.1 also includes improvements to several support objects. With AppWorx 5.1 you can:

• Define agent groups, applications, program types, libraries, spoolers, and thread schedules as role controlled objects.
• View a whole year in one screen when defining calendars.
• Select standard, OAE, or SAP when defining agents to determine which fields are displayed and automatically assign default values where appropriate.
• Skip all jobs on a particular agent with the Inactive feature.
• Run jobs on every agent in an agent group with the Multi-Execution feature.

Expanded and Simplified Export/Import Capabilities

With the 5.1 Export utility you can:
• Export all objects except agents, agent groups, logins, and users.
• Easily view reference objects when exporting.
• Export a chain without exporting its module details (chain components). With this feature you do not accidentally overwrite module definitions on the destination machine.

Improvements to AppWorx Explorer

From the 5.1 Explorer window you can:
• Enter information about the processing of a job with operator logs. You add, view, and query operator logs from the Backlog or History.
• Make changes (such as inactivating queues or starting agents) to more than one agent or queue at a time.
• View logs for all processes associated with each agent.

Improved Log Retention and Rollover

You can modify variables for the master's log files to control their:
• Log retention days: The number of days a log file is retained.
• Rollover interval: How many minutes a new log file is created.

Drag and Drop Support for AppMaster

AppMaster now supports a drag and drop feature for moving icons.

New AppWorx Command Line Interface Functions for Palm Devices

New commands have been added to the AppWorx Command Line Interface (CLI) for use with palm devices.

Improvement to the Oracle Applications Extension

AppWorx now uses an Oracle supplied database procedure, eliminating the need for the CONCSUB command. As a result, OAE is now faster, requires less environmental setup, and can run on NT.
improvements from version 4.6 to 5.0.x

the following improvements were made to appworx 5.0.x. they are relevant if you are upgrading from version 4.x.

job management capabilities—the job management capabilities were expanded:

• you can issue the reset, hold, and kill commands against two or more jobs at the same time.
• issuing the delete, reset, hold, and kill commands against a chain affects all children (chains and modules) of the chain.
• the incremental refresh performance was improved. it only reloads data for jobs that have changed instead of all jobs in the backlog.
• you can customize the backlog and history for a session by changing the order and size of the columns.
• a tree-structured forecast makes it easier to review scheduled operations.
• you can filter backlog and history by a particular agent and/or queue.

performance enhancements—performance was improved:

• imports and exports are over 10 times faster.
• command line execution is 30 times faster with the new awrun process than with the old sorun process.
• local and remote agents now share the same pool of parallel transaction server processes (awserver), greatly improving throughput. servers processes are initiated on an as-needed basis.
• job throughput was increased by new parallel processes (awexe and awprint) that handle job completion and printing. new processes are started for each job, off-loading the work from the appworx agent and single printer process.
• job throughput was increased by faster chain checking logic (conditions, threading).
• the server processes are smaller. they start faster and use less system resources.

chains—the following changes were made to chains:

• when a module that is part of a chain is in the backlog, you can switch its queue and appworx will maintain the module's thread setting, thereby maintaining the chain's integrity.
• multi-threaded chains are processed the same as multi-threaded modules.
• chains are retained in the backlog until all their children are complete.
• taking actions on running (initiated) chains will affect all children of that chain. for example, if you put a chain on hold, all modules in the chain will be put on hold.
File viewer—the file viewer was improved:
  • Faster performance than version 4.6.
  • A new print preview option was added.
  • Support for large text files.

AppMaster—support was added for the AppMaster software extension. AppMaster is used to monitor multiple AppWorx masters from a single workstation. For more information on AppMaster, contact your AppWorx account representative.

AppWorx command line interface—the AppWorx Command Line Interface (CLI) was added to manage masters, agents, queues, processes, substitution variables, and jobs from the command line. You can control access to the AppWorx CLI at the UNIX and AppWorx user levels.

Simpler network communications architecture—communications processes have greater resiliency to network and database disruption. Several communication processes were added to the master and remote agents to improve response time and reliability.

Time zones—a time zone can be selected for each schedule, giving you better control over job execution times.
What’s New with AppWorx 5.1
Section I

Introduction and Getting Started

Chapter 1: Introduction to AppWorx ................................................................. 3
Chapter 2: Using the AppWorx Desktop .......................................................... 21
1

Introduction to AppWorx

1.1 A Modular Approach to Operations ................................................................. 4
1.2 Overview of AppWorx Objects ......................................................................... 6
1.3 How AppWorx Works ......................................................................................... 8
1.4 Guidelines for Starting Out with AppWorx ....................................................... 10
1.5 Starting Out with Existing Scripts ..................................................................... 12
1.6 AppWorx License Options ................................................................................ 14
1.7 Naming Convention Guidelines ...................................................................... 16
1.8 Contacting AppWorx Technical Support ......................................................... 18
1.1 **A Modular Approach to Operations**

AppWorx replaces scripts with an object-oriented approach to operations. AppWorx provides a complete background-processing management system. With AppWorx, you can do all the traditional batch management tasks, as well as submit jobs and view output.

However, instead of writing scripts to perform tasks, you create AppWorx modules that run programs. You combine modules to create chains. You create a schedule for each chain (or for each module if you wish) using a full complement of scheduling tools including custom calendars and execution conditions.

As with all modular systems, you create individual objects once, then use them in different combinations to accomplish a variety of tasks. In the context of batch management, this can save you hours of time compared to writing, debugging, and maintaining scripts. A diagram of the AppWorx modular approach to operations is shown in Figure A.

**A Different Way of Thinking**

In a traditional operations environment, scripts drive operations. Scripts incorporate the information required to run one or more programs on a set schedule, direct output, and handle exceptions. The problem with scripts is the required time maintaining them when the system changes. For example, if a printer definition changes, you must change it in every script.

AppWorx takes the individual components of a script such as programs, schedules, printers, and variables, and lets you define them as discrete objects. You can then combine the objects in an unlimited number of combinations to handle your operations. The advantage is being able to change an object in one place, and have the changes roll over to every use of the object.
AppWorx uses a modular approach to replace the tedious tasks of writing and maintaining scripts.
1.2 Overview of AppWorx Objects

You use a variety of objects to accomplish your work in AppWorx.

With AppWorx you can use a variety of objects to control the jobs you run. For information on adding objects, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects. The objects listed below are presented in the order that they appear in this guide.

**Modules**—the basic building block in AppWorx. For each program you want to run (for example: FTP, application, database load), you must create a module. A module contains all the information required to execute a program and handle its output. Modules are run both individually and as components of AppWorx chains. Furthermore, a module can be a component of as many chains as you wish. If you change a module definition, the change is applied to every chain that includes it. For more information see chapter 8: Creating Modules.

**Applications**—logically group modules and chains. For more information, see topic 8.4.2 Defining Applications.

**Libraries**—define the path for the program a module runs. For more information, see topic 8.4.3 Defining Libraries.

**Program types**—define how programs accept input and handle output. For more information, see topic 8.4.4 Defining Program Types.

**Chains**—the AppWorx equivalent to job streams. Chains can include one or more components (modules and other chains), general scheduling information for the chain, specific scheduling information for each of its components, and conditions that must be met for each component to run. For more information, see chapter 9: Creating Chains.

**Data types**—specify the format for data passed to programs. They can include SQL statements that return values from your database. For more information, see topic 11.5 Defining Data Types.

**Calendars**—define groups of days, such as holidays, that you can use for schedules. Modules can be scheduled to run on, or to skip, the days in a calendar. For more information, see topic 10.6 Defining Calendars.

**Substitution variables**—store values that can be used in modules and chains. AppWorx lets you use substitution variables, such as #today, in prompts and execution conditions assigned to modules. For more information, see chapter 14: Defining Substitution Variables.

**Users**—control access to AppWorx. You can assign names, access permissions, user options, and roles to AppWorx users. For more information, see topic 16.2 Defining Users.
Roles—control access to all areas of AppWorx. In a traditional system, you create groups of users, printers, and applications. In AppWorx, roles replace groups. You can define roles for users, printers, and applications, as well as any other set of objects. Roles can contain any combination of objects, and objects can be assigned to any number of roles. For more information, see topic 16.4 Working with Roles.

Role authorities—control user access to AppWorx windows and give users add privileges for objects. You assign role authorities to your roles. For more information, see topic 16.4.3 Understanding Role Authorities.

Logins—allow operators and programmers to run programs that access a database or host without having to know the login and password. For more information, see topic 16.6 Defining Logins.

Agents—an instance of AppWorx, an agent is installed on each machine where jobs are executed. An agent can be a master's local agent, or a remote agent. The master schedules and controls job execution on all the agents assigned to it. For more information, see chapter 17: Working with Agents.

Agent groups—distribute the load between agents on one or more machines. For more information, see chapter 17: Defining Agent Groups and Assigning Agents to Them.

Queues—control the flow of jobs. All jobs must pass through an AppWorx queue to get to be executed. You control queue throughput by assigning a queue to a thread schedule. You can define an unlimited number of queues. For more information, see chapter 18: Administering Queues.

Thread schedules—schedules that are assigned to one or more queues or agents. Thread schedules define the number of concurrent jobs that can run through a queue or agent at different times of the day. For more information, see topic 18.3 Defining Thread Schedules.

Printers—define any output device including printers, faxes, and email. You can define a single printers, or through the use of distribution lists, multiple printers. For more information, see chapter 15: Defining Output Devices.

Printer groups—define organizational classes of output devices. When you define AppWorx printers, you assign them to one or more printer groups. For more information, see topic 15.4 Defining Printer Groups.

Print spoolers—an interface between AppWorx and an output device. For more information, see topic 15.5 Defining Print Spoolers.
1.3 How AppWorx Works

AppWorx uses object definitions stored in a database to give it an advantage over all other schedulers.

AppWorx is object-oriented. When you create an object, the information is stored in an AppWorx database. It is the reliance on object definitions stored in a database that give AppWorx its advantage over all other schedulers. A representation of how AppWorx runs jobs is shown in Figure A.

**Master and Agents:** The AppWorx master resides on a server and launches jobs locally via its local agent and remotely via remote agents. In AppWorx, a job represents a single program running on a server.

**Modules:** For AppWorx to run a program, you must define a module. A module includes information required for AppWorx to run the program such as program location and parameters.

**Chains:** You combine modules (and other chains) to create chains. A chain runs any number of programs and serves the same purpose as a traditional script. Instead of running a script, you run an AppWorx chain. Chains include scheduling and exception handling information.

**Chain Components:** Modules and chains assigned to the chain.

**Object Storage:** Chains, along with all other object definitions, are stored in the AppWorx relational database. This makes it easy to update an object’s definition, and have AppWorx apply the updates everywhere you use the object.

**Running Jobs:** The AppWorx master determines which modules and chains to launch by reviewing scheduling information in the database. When AppWorx launches a job, it sends it to the Backlog. The job remains there until it is processed. The job is sent to the appropriate agent/server where it is executed. You can also submit modules and chains to the Backlog using the Requests window or the AppWorx Command Line Interface.

**Monitoring and Reporting:** AppWorx monitors the progress of jobs, reporting the status. After a job completes executing, AppWorx captures the output. You can view the output online or print the output to a local Windows or system printer.
Figure A. How AppWorx processes jobs
1.4 Guidelines for Starting Out with AppWorx

You can get started with AppWorx by creating modules and defining simple objects as they are needed.

There are a number of different ways to get started with AppWorx. If you are very organized, you can begin by defining most of the objects you will use to build modules and chains, then build the modules and chains. Or you can begin by building modules, and defining the other objects as you need them. How you proceed is up to you. Both approaches will get you where you want to go. The basic steps for beginning with creating modules are outlined below.

Step 1: Create Modules

In AppWorx, a module runs a single program. To create a module, you specify a library, application, program type, program, queue, login, and printer. At the heart of the module is the program or job it runs.

What modules do you need to create? A good starting point is to create a module for each program in your existing shell scripts. If you do not have shell scripts, create a module for each program you want to run.

Remember: You use define modules once, but can assigning them to multiple chains.

Step 2: Define Objects

As you define modules, you use a variety of simple and compound objects. A compound object is made up of one or more simple objects. You can define objects as needed. Examples of simple and compound objects are listed below.

<table>
<thead>
<tr>
<th>Simple objects</th>
<th>Compound objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications</td>
<td>Modules</td>
</tr>
<tr>
<td>Libraries</td>
<td>Chains</td>
</tr>
<tr>
<td>Program types</td>
<td>Queues</td>
</tr>
<tr>
<td>Data types</td>
<td>Printer groups</td>
</tr>
</tbody>
</table>

Step 3: Create Chains

After creating modules, you can combine them to create chains. An AppWorx chain is equivalent to a script that runs many jobs. You can create schedules for running the chain as a whole, and also for scheduling each component of the chain.
Step 4: Define Queues

All jobs submitted to your system from AppWorx must pass through an AppWorx queue. A queue is a pipeline that can be used to control the flow of jobs to the server. By defining the appropriate set of queues, you can set a limit on the number of jobs that can be processed at any one-time for different periods during the day.

When you define a queue, you assign the queue a thread schedule. The thread schedule sets the number of concurrent jobs that can be processed by the queue. The number of jobs can be different for different times of the day.

Step 5: Set Up Roles to Control Access

After creating modules and chains, you create roles to control access to AppWorx. Think of a role as a container into which you place objects, including users. Users have access to all objects in the container or containers where they have been placed. Users, and other objects, can be assigned to any number of roles.
1.5 Starting Out with Existing Scripts

To replicate the functionality of an existing script, create an AppWorx module for each program run by the script, then add the modules to an AppWorx chain.

If you have been using scripts to run your operations, you can begin by taking one of the scripts and replicating its functionality in AppWorx. In AppWorx, a chain is equivalent to a script. Like a script, a chain runs one or more programs on a schedule. Unlike a script, AppWorx lets you pull together the necessary components from the AppWorx database to create a chain instead of typing all the information into a long script. The function of a script that may have taken hours to write can now be emulated in AppWorx in a few minutes.

Below is a description of the steps you take to turn an existing script into an AppWorx chain.

Procedure

To take a script and turn it into an AppWorx chain:

1. Identify all the programs run in the script.
   Programs might transfer files, load data, run reports, or run statistics.

2. For each program, create a module. For more information, see chapter 8: Creating Modules.
   With modules, you specify where the program is located, the name of the program, and where the output should be printed.
   If a program requires input, define a prompt for each parameter.
   You can specify scheduling information for the module or add it to a chain. In this example, we will add it to a chain, which in turn will be scheduled.

3. Save each module definition.
   Each module you have defined can be used in any number of chains.

4. Create a chain, adding the modules you created. For more information, see chapter 9: Creating Chains.
   The chain links the modules together and executes them in the order you specify. A chain accomplishes the same thing as a script.

5. Next, you can:
   • Add scheduling information to the chain by choosing the time and days you want the chain to run.
   • If appropriate, add scheduling information to each module in the chain.
   If appropriate, add predecessor statements and conditions under which each module will execute.
6. Save the chain.

After saving the chain, AppWorx will begin running it on the defined schedule.
1.6 AppWorx License Options

Your license key determines whether several ad-on features are included with your AppWorx master/agent instance.

Your license key determines whether several ad-on features are included with your AppWorx master/agent instance. AppWorx can be installed in one of three ways:

- Without Explorer
- With Explorer
- With the Graphical Analysis Package (which includes Explorer)

Each choice is described below.

**Without Explorer**

A license key without Explorer includes:

- The Jobs window, it is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog.
  For more information on the Jobs window, see topic 5.3 Using the Jobs Window.
- The basic Chain Summary feature including:
  - An unfiltered list of chain components, as well as the components belonging to all parent chains and child chains.
    When viewing a Chain Summary like this, you determine a job’s chain by scrolling through the Chains column.
  For more information on Chain Summaries, see topic 6.2 Viewing Chain Summaries.

**Explorer**

A license key with Explorer includes:

- The Explorer window replaces the Jobs window, it is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog. It also includes an object tree used to view a focused display of the Backlog.
  For more information on the Explorers window, see topic 5.2 Using the Explorer Window.
- The Explorer Chain Summary feature including:
  - An object tree.
  - A filtered list of chain components selected in the object tree.
  For more information on Chain Summaries, see topic 6.2 Viewing Chain Summaries.
Graphical Analysis Package

A license key with the Graphical Analysis Package includes:

- The Explorer window replaces the Jobs window, it is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog. It also includes an object tree used to view a focused display of the Backlog.
  
  For more information on the Explorers window, see topic 5.2 Using the Explorer Window.

- The Graphical Analysis Package Chain Summary feature including:
  
  - An object tree.
  - A filtered list of chain components selected in the object tree.
  - A Gantt chart used to show job execution statuses and times.
  - A Graphical flowchart of all jobs in the chain and their children.
    
    You can expand and collapse the chains in this view and select a job to bring it into focus in the list and on the tree. You choose whether to have AppWorx display predecessor relationships, execution order, and conditions.
  
  For more information on Chain Summaries, see topic 6.2 Viewing Chain Summaries.

- The Graphical Forecast feature.
  
  For more information on the Graphical Forecast feature, see topic 6.7 Viewing a Graphical Forecast.

- Chain Flows feature used to simulate chain execution when building chains.
  
  For more information on this feature, see topic 9.5 Viewing Chain Flows.

For information on enabling Explorer, see topic 2.7 Editing User Settings.

Upgrading to Explorer or the Graphical Analysis Package

To upgrade to Explorer or the Graphical Analysis Package, call AppWorx Technical Support at 1-877-APPWORX.
1.7 Naming Convention Guidelines

The initial implementation will require the creation of a comprehensive set of naming convention guidelines for AppWorx objects, particularly modules, chains, and aliases for chain components.

The best practice recommendation is to build upon the familiarity of an existing naming convention, rather than create a scheme entirely from scratch. An existing scheme provides welcome familiarity and can be expanded to allow for new circumstances.

Naming Conventions that Work

- Choose names that reflect the basic function of the object. The module name FTP is very good.
- Strive for generic names rather than specific names. This works well in situations where general objects of a utilitarian nature are built rather than building another object for every single different situation. Again, FTP is a good name for a module. COMPRESS_EXTENTS might be another.
- Login object names should take advantage of the commenting facility in AppWorx to clarify where the login is going to connect and to avoid identically named objects (which aren’t allowed in AppWorx anyway). For example, a login object can be named ‘fred@db1’. The @ symbol and all text after it are treated as a comment by AppWorx. For login purposes, only ‘fred’ is used as the user name.
- It is often more clear to those monitoring AppWorx if module and/or chain names include some type of ownership designator. For example, a chain named NETBACKUP_OPS might be identified as an operations group chain.

Naming Conventions that don’t Work

- Avoid existing naming conventions if they are clearly unworkable or create many names that are very similar in appearance. For example, chains called ‘MMN200A’, ‘MNN200A’, and ‘MNM200A’ can be very difficult to distinguish between.
- Avoid putting the same (obvious) prefix on the front of many object names (for example, ORA_COPYFILE, ORA_MOVEFILE, and ORA_RMFILE). In a list of many objects, lots of names with the same first few characters can be very difficult and slow to read.
- Avoid adding any time qualifier to object names. For example, MONTHLY_TRANSFER may work fine for now but what happens if the transfer frequency changes to daily or monthly? This would require a name change to the object or defining another object with a new name. Just plain TRANSFER might be better.
- Do not name modules ‘SCHED-<chain name>’, unless they are schedule modules. Standard modules with these names will not run.
1.8 Contacting AppWorx Technical Support

AppWorx Technical Support is available from 6:00 A.M. to 5:00 P.M. Pacific Standard Time, Monday through Friday.

If you encounter problems with AppWorx, you should make every effort to solve the problems using:

- The instructions provided in this manual.
- The knowledge base available at the AppWorx Support Site: http://support.appworx.com
  You can access the AppWorx Support Site from the AppWorx desktop by going to the Help menu and selecting AppWorx Support.

If you are not able to resolve the problem, call AppWorx Technical Support.

AppWorx Technical Support is available from 6:00 A.M. to 5:00 P.M. Pacific Standard Time, Monday through Friday. Emergency (24 by 7) technical support is available. Contact your AppWorx Account Manager if you are interested in purchasing emergency support.

How to Contact Technical Support

You can contact AppWorx Technical Support by:

Phone: 1-877-APPWORX (277-9679)
Fax: 425-644-2266
Email: support@appworx.com

Before You Call

Before you call AppWorx Technical Support, please have the following information available:

- Version number of AppWorx you are running
- System on which AppWorx is running (e.g. Sun, Hewlett-Packard, Windows)
- Operating system host name
- Operating system login information for the AppWorx account(s)
- Login information for the AppWorx account(s)
- Problem reference number if you are making a follow-up call on a previous problem
What to Expect

If you are calling AppWorx Technical Support for the first time, please be prepared to provide the following information:

- Your name
- Company name
- Location
- Phone number with area code

The AppWorx Technical Support representative will give you a problem reference number. Please write down the number. If you call again about the same problem, the number will allow the representative to more quickly access the history of the problem.
2.1 Introduction to the AppWorx Desktop

From the AppWorx desktop you can access windows for all AppWorx operations, administration, and options.

If you are using a Windows client with 800x600 resolution, you will need to select the Windows auto hide taskbar option.

From the desktop you can:

• Define AppWorx objects such as modules and chains.
• Run jobs.
• Monitor operations.
• View output.

ToolTips

ToolTips provide a brief description of AppWorx buttons, icons, and fields. To see a ToolTip, rest the mouse pointer over the button, icon, or field. A ToolTip appears after the
mouse pointer has remained motionless for a second or two. In Figure A, the mouse pointer is resting on the Library icon in the toolbar. You can disable ToolTips by going to the View menu and unchecking the ToolTips option.

**Toolbar**

The toolbar consists of a row of icons on the top of the screen. When these icons are clicked, AppWorx opens a corresponding window. You can view or hide the toolbar by opening the View menu and checking the toolbar option. You can add to or edit the icons displayed on the toolbar with the Options menu by selecting Settings. For instructions on editing user settings, see topic 2.7 Editing User Settings.

**Taskbar**

The taskbar is a graphic bar on the desktop used to select active AppWorx windows.

From the taskbar, you can right-click to:

- Restore a window to the desktop or minimize it to the taskbar.
- Maximize a window to fill the desktop.
- Move a window to the front of the desktop.
- Close a window.

Selector windows are special windows used when defining AppWorx objects. They are not displayed on the taskbar because they do not contain unique information and are represented by icons on the toolbar.

**Status Bar**

The status bar is displayed across the bottom of the Explorer (or Jobs) window. Its color alerts you to the status of the AppWorx master, agents, and jobs running in the Backlog. When the Explorer (or Jobs) window is minimized it uses the same color scheme on the taskbar. For more information on the status bar, see topic 5.4 Monitoring with the Status Bar and Object Icons.

**Closing All Windows or Selector Windows**

To close all windows, go to the View menu and select Close all. To only close the selector windows, select Close selectors. For more information on selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

**Bringing Error Dialogs into View**

If you minimize the AppWorx desktop when an error message is displayed, the AppWorx desktop may seem to lock up. This is because the Error dialog is out of view. To bring the error dialog back into view, hold down the Alt key and Enter the Tab key until you highlight the Java coffee cup icon.
2.2 Working in the AppWorx Windows

In many of the windows the columns can be sorted and rearranged. AppWorx conforms to most GUI standards, including keyboard navigation.

### Sorting Columns

You can display the items within a column in a preferred order. Click the header of a desired column to view its modules in descending order. Click a second time to view the modules in ascending order. Notice that the arrow to the right of the column name reflects this change. Some columns can be clicked a third time to display the modules in their default order. The arrow may be displayed in a separate column, or not at all when modules are in their default order. You can click the columns additional times to cycle through the options.

### Changing Column Order

To change the order of the columns, use the mouse to point to the column you want to move, hold down the mouse button, and drag the column to the new position (see Figure A).

**Note:** Headings can be renamed. For information on renaming headings, see your AppWorx administrator.

![Figure A. You can change the order of the AppWorx columns.](image-url)
Assigning Options

Many windows, such as the Users window shown in Figure B, allow you to assign multiple options.

To select a single option you can double-click it. To assign one or more options, select the values you want to use. The table below describes how to move user options between Assigned and Unassigned boxes.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move a selected value between the two columns</td>
<td>Double-click the value.</td>
</tr>
<tr>
<td></td>
<td>-or- Select the value and click the single arrow button.</td>
</tr>
<tr>
<td>Move all values between the two columns</td>
<td>Click on the double arrow button.</td>
</tr>
<tr>
<td>Make multiple contiguous selections</td>
<td>Hold down the Shift key and click the first and last values.</td>
</tr>
<tr>
<td>Make multiple nonadjacent selections</td>
<td>Hold down the Control key and click on the values.</td>
</tr>
</tbody>
</table>

To select an option, you can double-click it. To move all options, select values and click on the double arrow button.

**Keyboard Navigation**

You can use the following keyboard navigation in AppWorx:

- Control+Shift Tab will set the focus on the current tab, and then the right & left arrow keys will navigate to the other tabs.
- If a table has focus, then Control Tab will move the focus out of the table.
- When you are in a table, Enter and Tab are used for navigation within the table.
- To expand an object’s key, press Enter.
- To select a hot key, click Alt+<the key>. 

*Figure B. Double click or use the arrow key to assign/unassign options.*
2.3 Launching AppWorx

To launch AppWorx for the first time, enter the AppWorx URL and select an option to either launch using Java WebStart or the Java Plugin. If you open AppWorx using WebStart a second time, you will be prompted to add icons to your start menu and desktop.

You must launch AppWorx before you can log on. There are three ways to launch AppWorx for the first time:

- Click on the AppWorx URL link sent to you via email by your AppWorx administrator.
- Select the AppWorx bookmark from your browser.
- Type the AppWorx URL into your browser's address window.

AppWorx displays the Introduction to AppWorx Operations screen shown in Figure A.

**Note:** If you are using Internet Explorer, this page will include a link to install Java2. With Netscape, you will automatically be prompted to install Java2 if you do not have it.

![AppWorx Introduction Page](image)

**Figure A.** From the Introduction to AppWorx page, you can start AppWorx or AppMaster with a browser or WebStart. You can also install WebStart if necessary.

From this page you can launch AppWorx in one of two ways:

- Using Java WebStart, to run the AppWorx desktop as its own application without a browser window (recommended).
- Using the Java Plugin, to run the AppWorx desktop within the browser window.

If you do not already have WebStart on your computer, there will be a link to install it.
When you launch AppWorx with Java WebStart a second time, you will be prompted to add icons to your desktop and start menu. With them, you can link directly to AppWorx without opening a browser window.

**The AppWorx Signed Certificate**

AppWorx provides a signed security certificate with its AppWorx Java client. By granting the certificate when you first launch the AppWorx Java client, you can bypass the need to set up an individual Java security policy file for each user. This certificate guarantees that the AppWorx applet code is tamper free.

When you first launch AppWorx, the Java Plug-in Security Warning window shown in Figure B is displayed. You are asked to either grant or deny a signed certificate provided by the AppWorx Corporation, asserting that the AppWorx application code is safe to download to your system.

It also allows the user to access clipboard functionality and user configuration files. If you choose to deny the certificate, you will have to modify the policy file permissions for each of your client users to use the clipboard for capturing information for support. Notice that the user may grant the certificate for the current session only, or choose to grant it always. If the certificate is granted always, the user will not have to go through this security check each time an AppWorx session is launched.

**Launching AppWorx from the Command Line**

If you are running on a console or through an X-Windows enabled terminal, you can type `appworxj` (or in Motif `startso client`) to launch AppWorx.

**Next Step**

Now you have launched AppWorx. The next step is to log on. For more information, see topic 2.4 Logging on to AppWorx.
2.4 Logging on to AppWorx

To log on to AppWorx, enter your user name and password.

Once you have launched AppWorx, you can log on. To log on to AppWorx:

1. At the Logon window shown in Figure A, enter your AppWorx user name and password, and select a master from the drop-down list.

2. To have AppWorx remember your user name and master, select the Remember logon button.

3. To accept the entered information and log on to AppWorx, click OK.

   Note: If you have a large number of output files or jobs in the Backlog, AppWorx will display a message alerting you that the display has been truncated.

Re-logging on to AppWorx

There may be times when you want to re-log on to AppWorx under a different user name or connect to a different master. To change your logon from the desktop, go to the File menu and select Re-Login.
Changing Your Password

To change your AppWorx password:

1. On the desktop, go to the Options menu and click Change Password.
   AppWorx displays the Change password window shown in Figure B.
2. Enter your current password.
3. Enter and Re-enter your new password.
4. Do one of the following:
   • To accept the change, click OK.
   • To disregard the change, click Cancel.
2.5 Adding, Editing, and Deleting AppWorx Objects

Add, edit, and delete objects using the selector windows. Modules and Chains selector windows include an Application box and a Copy function.

You add, edit, and delete objects using the selector windows.

To open a selector window, do one of the following:

- Select an icon from the toolbar on the desktop (see Figure A).
- Select an item from the Objects Admin menu on the desktop.
- Select the icon next to an object’s field when defining another object (see Figure B).
- Enter the corresponding hot key.

Note: AppWorx roles control access to the selector windows. If you do not have access to one, see your AppWorx administrator.

Adding and Editing Objects

To add, edit, or delete an AppWorx object:

1. Open the selector window that corresponds to the object you wish to add or edit.
   
   Only the objects assigned to you via roles will be displayed in the selector window. In Figure B, the Queues selector window is open.
   
   The Modules and Chains selector windows include an Application box. Applications specify a group of modules and chains. The application you select determines which modules and chains are listed in the table. For more information on applications, see topic 8.4.2 Defining Applications.
   
   Note: Selector windows are not displayed on the taskbar because they do not contain unique information.
   
2. To update an existing object, select the object and click Edit.
   
   If you are editing an object, you can type the first few letters of its name in the Search field, and AppWorx will find it.
   
   The Search field accepts valid UNIX regular expressions. For information on syntax accepted by regular expressions, see Appendix D: Regular Expression Tables.
   
3. To define a new object, click New.
AppWorx opens a window for the object. If you are defining a new object, the fields will be empty, except where defaults come preselected.

Required fields are marked with a '*' symbol to their right.

4. Enter values for the fields on the various tabs of the object.

When you enter or change a field value, AppWorx displays a triangle next to the current tab’s label to signify unsaved changes. A red triangle signifies unsaved changes made to a sub-element of a tab. For more information, see topic 8.3 Updating Unsaved Changes.

5. If you wish, you can select the Roles tab to assign the object to one or more roles. For more information on assigning objects to roles, see topic 16.4.4 Assigning Roles to an Object.

If you are assigned to a Maintenance role, the object will automatically be assigned to your role. You can view a list of objects that you have access to from the View menu by selecting View Assigned Objects. If you do not have access to the object, see your AppWorx administrator.

6. Select the appropriate button:
   • To save the object’s definition and close the window, click OK.
   • To add/update the object’s definition and keep the window open, click Save.
   • To close the window without updating the object’s definition, click Cancel.

Deleting Objects

To delete an object, highlight the object on its selection window and click Delete.

If the object you are deleting is referenced (used) by one or more objects, you must remove the references before it can be deleted. If you try to delete an object without first removing the references, AppWorx will display a message saying it is in use. For information on viewing references for an object, see topic 2.6 Viewing Object References.

If modules or chains are in the Backlog, you cannot delete their definitions. You will need to wait until they complete executing.

Copying Modules and Chains

The Modules and Chains selector windows include a Copy function. For more information, see topics 8.9 Copying Modules and 9.7 Copying Chains.
2.6 Viewing Object References

You can view all the references of an object by selecting an object and clicking the Usage button.

If you wish to delete an object, you must remove its references first. You can view all the references of an object by selecting an object and clicking the Usage button.

Figure A. To view all references for an object, click Usage on any object selector window.

Procedure

To view all references for an object:

1. Open the selector window for an object type. There are several ways to open an object selector window:
   - Select an icon from the toolbar on the desktop.
   - Click the icon next to an object’s field when defining another object.
   - Select an item from the Objects Admin menu on the desktop.
   - Enter the corresponding hot key.

   AppWorx opens an object selector window for the object type. In Figure A, the Printers selector window is opened.

2. Highlight an object in the table and click Usage.

   AppWorx opens the References window for the object (see Figure A). Each instance where the object is referenced is listed by name and object type.
2.7 Editing User Settings

To set the AppWorx user settings, select Settings from the Options menu. You can apply these settings to the current session, or save them for future sessions.

You can edit and save general and toolbar user settings.

Editing General Settings

To set the AppWorx user sessions:

1. From the Options menu, choose Settings
   
   AppWorx displays the General tab of the Settings window in Figure A.

2. To adjust the settings on the General tab, reference Table A.

3. If you wish, you can also edit the toolbar display at this time (see below).

4. To save the settings for the current session and future sessions, click OK. To call up the settings that were last saved, click Restore.

   Note: To enable/disable Explorer you must re-log into AppWorx.

Editing the Toolbar Display

You can customize the toolbar by adding and removing icons. To add or remove icons on the toolbar, select the toolbar tab shown in Figure B. Double click or use the arrow keys to move items between the Unassigned and Assigned columns. For details on assigning/unassigning values, see topic 2.2 Working in the AppWorx Windows.
Table A. AppWorx User Settings

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Monitoring</td>
<td>When the Active box is checked, continuous monitoring is activated, and AppWorx refreshes the Backlog, History, Agent Summary and Queue Summary at the number of seconds set in the Frequency (seconds): field. In Figure A, continuous monitoring is turned on, the refresh frequency is set to 10 seconds (the lowest number of seconds available).</td>
</tr>
<tr>
<td>• Active</td>
<td></td>
</tr>
<tr>
<td>• Frequency (seconds):</td>
<td></td>
</tr>
<tr>
<td>Explorer</td>
<td>This check box enables Explorer. Explorer is an optional add-on that offers an alternative method for tracking/managing jobs. For more information on Explorer, see topic 5.2 Using the Explorer Window, and topic 1.6 AppWorx License Options. Note: To enable/disable Explorer you must re-log into AppWorx.</td>
</tr>
<tr>
<td>• Enabled</td>
<td></td>
</tr>
<tr>
<td>Job History Limits</td>
<td>You enter a number in the Maximum rows field to control how many rows will be displayed in the History. In Figure A, the maximum rows are set to 500. You can also determine how long the records will be displayed using the Previous minutes field. For example, if you set Previous minutes to 60, only jobs that have been run within the last 60 minutes will be displayed. In Figure A, the Previous minutes field is set to 30.</td>
</tr>
<tr>
<td>• Maximum Rows:</td>
<td></td>
</tr>
<tr>
<td>• Previous minutes:</td>
<td></td>
</tr>
<tr>
<td>Use database time zone</td>
<td>If this check box is selected, all times are shown in database time. If it is not selected, then the time in the Submit window when submitting a job, will be local time (the time set on the client machine). This is translated to database time for running the job. Times shown in the Backlog, History, on the status bar, and in the lower right corner of the screen are always in database time.</td>
</tr>
<tr>
<td>Auto start Explorer</td>
<td>When this box is checked, AppWorx automatically opens the Explorer window when you log in.</td>
</tr>
<tr>
<td>Current session only</td>
<td>When this box is checked, AppWorx applies the changes you make to the current session only.</td>
</tr>
</tbody>
</table>

What are User Options?

User options are additional settings selected by your AppWorx administrator. They control user access to AppWorx features such as the Hide feature in the Output window. For more information on user options, see topic 16.3 Setting User Options.
2.8 Linking to Online Manuals, AppWorx Support, and the About AppWorx Window

To link to the online manuals, the AppWorx Support Site, or the About AppWorx window, select the appropriate item from the Help menu.

The AppWorx desktop offers links several help resources.

From the Help menu, you can:

- View the AppWorx Online Manuals.
- Connect to the AppWorx Corporate Web site
- Connect to the AppWorx Support Web site
- View the About AppWorx window

Viewing the Online Manuals

Use the AppWorx Manual option to view the AppWorx manuals in the JavaHelp viewer. When you select the AppWorx Manual menu item, AppWorx opens the AppWorx Help window shown in Figure A. Many of the windows include context sensitive Help buttons that link to a corresponding topic in the AppWorx manuals.

PDF files for all AppWorx manuals are available on the AppWorx product CD and the AppWorx Support Site.

Connecting to the AppWorx Web Sites

Use the AppWorx Web Site option to access information on AppWorx including our products and services. Use the AppWorx Support link to open new support issues, check the status of open support issues, browse our Frequently Asked Questions, view announcements and holiday information, view Release Notes, browse our Knowledge Base and read the current and back issues of our newsletters.

When you select the AppWorx Web Site or AppWorx Support menu items, AppWorx launches a Web browser and opens the appropriate site.
Viewing the About AppWorx Window

Use the About AppWorx window to view details on your AppWorx build, this information may be used when calling AppWorx Support. When you select the About AppWorx menu item, AppWorx displays the About AppWorx window shown in Figure B.

The About AppWorx Operations window includes:

- The AppWorx release number
- Workstation Java VM information
- Server Java VM information
- Java VM free memory information
- The DNS name and IP address of the computer you are using
- The RMI host’s DNS name and IP address
- The RMI host’s port number

Click **Copy** to place the system information on your clipboard.

Click **Versions** to view a list of file names and versions in the File Versions window. This window also includes a **Copy** button. An AppWorx Technical Support Specialist may ask you to copy and email this information when troubleshooting.
Chapter 2: Using the AppWorx Desktop
Section II

Operations

Chapter 3: Submitting Requests ................................................................. 41
Chapter 4: Viewing and Printing Output ................................................. 53
Chapter 5: Monitoring and Managing Jobs ............................................ 67
Chapter 6: Viewing Summaries and Forecasts ...................................... 119
Chapter 7: Using the AppWorx Command Line Interface ....................... 139
3

Submitting Requests

3.1 Introduction to Submitting Requests ................................................................. 42
3.2 Requesting Modules and Chains ................................................................. 44
3.3 Editing and Submitting Modules and Chains ........................................ 46
  3.3.1 Responding to List of Values (LOV) Prompts ......................................... 48
  3.3.2 Responding to Multiple Select (MS) Prompts ........................................ 50
3.1 Introduction to Submitting Requests

You can submit modules and/or chains on an ad hoc basis. After submitting one or more modules/chains, you can view their status from the Jobs/Explorer window.

There may be times when you want to run jobs outside of a set schedule. You can submit modules and chains on an ad hoc basis.

Figure A. Select one or more modules or chains and click Request, or simply double-click a single module or chain to request.

Requesting and Submitting Modules and Chains

The ability to request modules and chain on an ad hoc basis offers a useful method for end users who want to request reports that have not been scheduled ahead of time.

When requesting modules and chains, you use the following two windows:

- The Requests window shown in Figure A is used to select one or more modules/chains.
- The Submit window shown in Figure B is used to complete prompts, select options, and submit the requested modules/chains to the Backlog.

When you request jobs, you can set the start date and time, queue, designated printer, printer option, number of copies, and, print function (LOG, PRINT, or STORE). You can also enter values to customize reports when prompts are defined for a module.
Checking Module Status

After submitting a module, you can view its status in the Backlog and History. For information on the Backlog and History, see chapter 5: Monitoring and Managing Jobs.

Viewing Output

After a module executes, you can view and print output online using the AppWorx File Viewer. The File Viewer is accessible from both the Jobs/Explorer and Output windows. For information on accessing the File Viewer from the Jobs/Explorer window, see topic 5.9.5 Viewing Job Output. For information on viewing and printing output, see chapter 4: Viewing and Printing Output.
3.2 Requesting Modules and Chains

To request modules or chains, open the Requests window, select one or more modules and/or chains, and click Request, or simply double-click to submit a single module or chain. The selected modules and chains are displayed in the Submit window where you can set options and parameters and submit them.

The Requests window offers a quick and easy way to request modules for submission.

![Figure A. The Requests window](image)

**Procedure**

To request one or more modules from the Requests window:

1. Open the Requests window shown in Figure A by going to the Operation menu and selecting Requests, or by selecting the Requests icon from the toolbar.

2. Select an application from the **Application** list box on the left side of the screen. Applications specify a group of modules and chains. The application you select determines which modules and chains are in the table. Only the applications and modules assigned to you via roles will be displayed.

   In Figure A, the BATCH application has been selected.

3. Click, Shift+Click, or Ctrl+Click to select one or more modules from the module list box on the right side of the screen.

   When selecting a module/chain, you can type the first few letters of its name in the **Search** field, and AppWorx will find it.

   The **Search** field accepts valid UNIX regular expressions. For information on syntax accepted by regular expressions, see *Appendix D: Regular Expression Tables.*
**Note:** You can double-click to quickly request a single module.

In Figure A, three modules are selected.

4. To request the module(s), click **Request**.

AppWorx opens the Submit window and displays the selected items as tabs (see Figure B).

Each module can be viewed by selecting its tab on the top of the window. In Figure B three modules have been requested: ORDERS_FTP, ORDERS_NATIONAL, and ORDERS_REPORT.

![Submit window](image)

**Figure B. The Submit window**

After a module/chain has been submitted, it remains in the Submit window until its tab is closed (by clicking the **Close** button) or the window is closed (by clicking the 'X' button on the title bar).

Modules that are requested multiple times will display on separate tabs. In Figure C, the ORDERS module has been requested twice.

![Module and chains](image)

**Figure C. Module and chains can be requested and displayed multiple times.**

**Next Step**

Now you have requested one or more modules and/or chains. The next step is to submit them. For more information, see topic 3.3 **Editing and Submitting Modules and Chains**.
3.3 Editing and Submitting Modules and Chains

You can set the prompts and options when you submit modules and chains. To respond to a prompt in a module, accept the default value if one is displayed, enter a value, or click the LOV or MS button to select value(s) from a list.

You can set the prompts and options when you submit modules and chains.

Procedure

To edit and submit a module/chain:

1. From the Submit window shown in Figure A, select the tab for the module you wish to submit.
   For information on requesting modules from the Requests window, see topic 3.2 Requesting Modules and Chains.

2. If prompts were defined for the module or chain, they are displayed in the Prompts table.
   There are several ways to respond to a prompt:
   • Accept the default value if one is displayed. 200.1.1.58 is the default value for the first prompt in Figure A. If you change the default value, you can select the Default button to bring it back.
   • Enter a value in the default column.
   • Click the LOV or MS button (if enabled) to pick a value from a list.
     The LOV button lets you select a single value. The MS button lets you select multiple values. The LOV and MS buttons will be enabled only if a list of values or multiple select prompt has been defined for the module.

3. If you wish, select a different printer, print option (if available), and number of copies.
   The printers displayed in the list are determined by the printer group assigned to the module.

Note: Print options allow users to request a specific variable, setting, address, or orientation. They are available if the printer you select is assigned to a spooler that includes them. For more information, see topic 15.6 Adding Print Options to Spooler Definitions.

Figure A. You can edit the fields for each module/chain and click Submit.
4. Select a queue from the Queue list box.

The Queue list box will be active only if you have been assigned the Request Queues user option by your AppWorx administrator. If you have not been assigned this option, the Queue list box will be read-only, and will list only the default queue of the module.

5. Select a print function from the Function drop-down box. There are three print functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG</td>
<td>The output will not print. It is available for online viewing on both the Output and Jobs/Explorer windows.</td>
</tr>
<tr>
<td>PRINT</td>
<td>The output prints at the designated printer and can be viewed from the Output and Jobs/Explorer windows.</td>
</tr>
<tr>
<td>STORE</td>
<td>The output is stored as a hidden file. Again, it can be viewed from the Output and Jobs/Explorer windows (through a query in the Output window).</td>
</tr>
</tbody>
</table>

6. To set a different start date and time, place the mouse cursor in the Start Date or Start Time field and enter a new date or time.

When entering start times, you can include time zone abbreviations. For more information on time zones, see topic 17.4 Setting the Master’s Time Zone.

If the date and time are set ahead, the job status will be shown as DATE_PENDING when the module/chain is inserted to the Backlog.

7. To submit the module, click Submit.

If the values assigned to prompts are invalid, AppWorx will display an error message.

You can follow AppWorx as it processes the job by watching the status messages displayed in the lower left corner of the Submit window. AppWorx also displays the JOBID assigned to the job. In Figure B, the status and JOBID (Job was Successfully submitted: JOBID = 40529) are displayed.

8. To remove a tab from the Submit window, click Close. Clicking the ‘X’ button on the title bar closes the window.

Subtopics

The subtopics that follow provide additional information on responding to a list of values and multiple values prompt:

3.3.1 Responding to List of Values (LOV) Prompts
3.3.2 Responding to Multiple Select (MS) Prompts
3.3 Editing and Submitting Modules and Chains

3.3.1 Responding to List of Values (LOV) Prompts

If you select a list of values prompt, AppWorx displays the LOV button. To respond to the prompt, do one of the following: accept the default value if one is displayed, enter a value, or click the LOV button.

If you select a prompt defined as LOV, AppWorx displays the LOV button. You can click the LOV button (or double-click the prompt) to display a list of values (see Figure A).

If you click the LOV button, AppWorx displays the List of Values window shown in Figure B. The values displayed will be different for each prompt. To select a value for this prompt, select the value in the list and click OK or double-click the value. Remember, an LOV prompt accepts only one value.
Figure B. List of Values window
3.3 Editing and Submitting Modules and Chains

3.3.2 Responding to Multiple Select (MS) Prompts

If you select a prompt that takes multiple values, the MS button will be displayed. To respond to a prompt that accepts multiple values: click the MS button. AppWorx displays the Multiple Selection Values window shown in Figure B. The values displayed in the Assigned column are the default values for the prompt. Additional values are displayed in the Unassigned column.

If you select a prompt that takes multiple values, AppWorx displays the MS button. You can click the MS button (or double-click the prompt) to display a list of multi select values (see Figure A).

To respond to an MS prompt:

1. Click the MS button.

   The Multiple Selection Values window displays as shown in Figure B. The values displayed in the Assigned column are the default values for this prompt. Additional values are displayed in the Unassigned column.
2. Select the values you want to use.
   For information on assigning options, see topic 2.2 Working in the AppWorx Windows.

3. To accept the values, click Close.
   AppWorx displays an ID number in the Default column of the Submit window. When AppWorx runs the module, it uses this number to retrieve the values for the prompt.

**Figure B.** Default values for an MS prompt are displayed in the Assigned column. Use the arrow buttons to assign/unassign values.
4

Viewing and Printing Output

4.1 Introduction to Viewing and Printing Output ............................................................. 54
4.2 Working with the Output Window ............................................................................. 56
4.3 Querying the Output Window for Specific Jobs ....................................................... 58
4.4 Viewing Job Output with the File Viewer ............................................................... 60
4.5 Printing Job Output ................................................................................................... 62
4.6 Using File Associations ............................................................................................ 64
Chapter 4: Viewing and Printing Output

4.1 Introduction to Viewing and Printing Output

Using the File Viewer, you can view output and print it to a local or system printer.

After you submit a job and it executes, you can view the output as plain text, HTML, or rich text format output using the File Viewer shown in Figure A. You can also use an alternate viewer using AppWorx file associations. For information on setting file associations, see topic 4.6 Using File Associations. Output can be printed to a local or system printer.

Opening the File Viewer Window

You can access the File Viewer from:

- The Jobs/Explorer window.
- The Output window.

The Jobs/Explorer window is used to monitor and manage AppWorx jobs, as well as agents and queues. To open the File Viewer from The Jobs/Explorer window, right-click a job in the History or Backlog and select Output from the pop-up menu. For more information on viewing output from the Jobs/Explorer window, see topic 5.9.5 Viewing Job Output.

The Output window provides access to job output. For users that do not have access to the Jobs/Explorer window. To open the File Viewer from The Output window, select a job and click the View button. For more information on viewing output from the Output window, see topic 4.2 Working with the Output Windows.
Querying for Jobs

You can query the History on the Jobs/Explorer window and you can query the Output window to search for specific jobs by module name, chain, agent, requestor, etc. For more information on querying, see topics 5.14 Querying for Jobs in the History and 4.3 Querying the Output Window for Specific Jobs.

Viewing Output

After a job has completed, you can view the output in the File Viewer as shown in Figure A.

You can also associate types of files with other viewers. For example, if you are generating an .xls file, you can have AppWorx automatically launch Microsoft’s Excel as the viewer. To do this, you must specify the association in the File Association window. For information on setting file associations, see topic 4.6 Using File Associations.

Printing Output

After viewing a report, you can preview your print job and print it to a local Windows printer or to a system printer. These options are available on the File Viewer’s File menu.
4.2 Working with the Output Window

The Output window provides AppWorx users access to job output without requiring that they have the necessary role access to monitor and manage jobs. They can search for specific jobs using the Query function, hide jobs listed using the Hide function, and update the display to see new jobs using the Refresh function.

With the Output window shown in Figure A, you can:

- Open the File Viewer to view and print job output
- Search for specific jobs using the **Query** function
- Hide jobs listed using the **Hide** function
- Update the display to see new jobs using the **Refresh** function

---

**Table:**

<table>
<thead>
<tr>
<th>Status</th>
<th>Job K</th>
<th>Module</th>
<th>User</th>
<th>Yr Date</th>
<th>Agent</th>
<th>Chain Name</th>
<th>Email</th>
<th>Printer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logged</td>
<td>120W</td>
<td>PRODUCTS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ALL RECV ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>122X</td>
<td>REPORT</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>123X</td>
<td>INVENTORY</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>125X</td>
<td>PROCESS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>126X</td>
<td>LOAD_DATA</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>127X</td>
<td>FTP_ORDERS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>128X</td>
<td>REPORT</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>129X</td>
<td>INVENTORY</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>130X</td>
<td>PROCESS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>131X</td>
<td>LOAD_DATA</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>132X</td>
<td>FTP_ORDERS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>133X</td>
<td>REPORT</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>134X</td>
<td>INVENTORY</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>135X</td>
<td>PROCESS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>136X</td>
<td>LOAD_DATA</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>137X</td>
<td>FTP_ORDERS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>138X</td>
<td>REPORT</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>139X</td>
<td>INVENTORY</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
<tr>
<td>Logged</td>
<td>140X</td>
<td>PROCESS</td>
<td>JACK</td>
<td>Apr</td>
<td>PROD1</td>
<td>ORDERS</td>
<td>161</td>
<td>TEST</td>
</tr>
</tbody>
</table>

---

**Figure A.** You can view and print output from the Output window.

**Querying for Specific Output**

To search for specific jobs, click **Query** on the Output window. You can query for a specific job by module name, requestor, printer, creation date, and output status. For more information on querying the Output window, see topic **4.3 Querying the Output Window for Specific Jobs.**
Hiding Jobs
You can hide job outputs by selecting one or more jobs and clicking the Hide button (see Figure A). This changes the status of the selected jobs to stored and removes the job from the viewable list.

To display hidden jobs, you must use the Query function and query for the STORE status. To select multiple contiguous jobs, use Shift+Click. To select multiple noncontiguous jobs, use Ctrl+Click.

If the Deny Outmgr Hide option was assigned to you by your AppWorx administrator, you will not be able to use the Hide button. For more information on user options, see topic 16.3 Setting User Options.

Refreshing the Display
When you first open AppWorx, the Output window will show the jobs that have completed. If other jobs complete while you have the Output window open, they will not automatically be added. To see recently completed jobs, click the Refresh button.

Logged Jobs Displayed
By default, only jobs whose output was ‘logged’ will be displayed in the Output window. These jobs have a logged status. You can see printed, stored, or viewed jobs by running a query and searching for jobs with one or more of these statuses.

Output and Roles
When you are setting up roles, consider setting up a role for end users that includes access to the Requests and the Output windows only. This lets end users submit, view, and print their specific job requests.

Output Restricted to User
By default, only the jobs you have submitted will be displayed in the Output window, unless the View all User Outputs option was assigned to you by your AppWorx administrator. With this option set, you will have access to all outputs for the modules you have been assigned access to regardless of the user that submitted the module. For more information on user options, see topic 16.3 Setting User Options.
4.3 Querying the Output Window for Specific Jobs

To search for specific jobs on the Output window, click the Query button. You can query by modules, chains, requestors, output functions, printers, agents, start times, and job IDs.

If the number of jobs listed in the Output window is overwhelming, or you wish to view output from jobs which exceed the settings for the **Job History Limits**, you can run an output query.

**Procedure**

To search for specific jobs:

1. From the Output window, click the **Query** button
   
   AppWorx displays the Output Query window shown in Figure A.

2. Select items from one or more of the labeled boxes/fields and click **OK**.
   
   The boxes/fields are described in Table A.
   
   Use Shift+Click and Ctrl+Click to select or deselect more than one entry in a list.
   
   To cancel search criteria and close the window, click **Cancel** in the Output Query window.

**Viewing Query Results**

AppWorx runs the search and displays the queried results. The **Apply Query** check box will be selected (see Figure B). To view the unqueried Output window, uncheck the **Apply Query** box. You can view the queried results again by rechecking the **Apply Query** box. If you click the **Query** button again, AppWorx returns you to the Output Query window.

AppWorx displays the search criteria you defined on your last query. To run a new query, select new data and click **OK**.
Table A. Output query options

<table>
<thead>
<tr>
<th>Query option</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>You can use the ‘_’ wildcard to represent a single character, and the ‘%’ wildcard to represent an unlimited number of characters. You can type the first few letters of the module’s name to search for it. <strong>Note:</strong> To query for a job that ran under an <strong>Alias</strong>, enter the alias name.</td>
</tr>
<tr>
<td>Chain</td>
<td>This option searches for job output for modules in chains. The chains themselves are not returned by queries because they produce no output.</td>
</tr>
<tr>
<td>Requestor</td>
<td>This list will contain only your user name unless you have been assigned the <strong>View all User Outputs</strong> user option by your AppWorx administrator.</td>
</tr>
<tr>
<td>Status</td>
<td>These options relate to the <strong>Output function</strong> set for the job: <strong>LOG:</strong> Output function for module was set to LOG and the output is available for printing, but has not been printed. Outputs with this status appear on the Output window list by default. <strong>PRINT:</strong> Output function was set to PRINT. <strong>STORE:</strong> Output function for the module was set to STORE. You must query to see outputs with this status. System outputs (‘o’ files) are typically stored.</td>
</tr>
<tr>
<td>Printer</td>
<td>In AppWorx, a printer can be any output device.</td>
</tr>
<tr>
<td>Agent</td>
<td>Choose from the agents that jobs ran on, not agent groups the jobs may have been assigned to.</td>
</tr>
<tr>
<td>From Date &amp; Time, To Date &amp; Time</td>
<td>You set the date and time by double-clicking in the desired field and selecting values from the Date Time Selector window shown in Figure A.</td>
</tr>
<tr>
<td>JobID</td>
<td>When using this field, include the decimal value where appropriate. Decimal values at the end of JobIDs indicate that jobs have been restarted. Every job has .01 added to its JobID each time it is restarted.</td>
</tr>
</tbody>
</table>

![Figure B. The Output window with the Apply Query check box checked](image)
4.4 Viewing Job Output with the File Viewer

You can access the File Viewer from the Jobs/Explorer window or the Output window.

After a job executes, you can view its output online. This can be useful if you are trying to debug a new program or if the output is such that you do not need a hard copy. You can enter page and line numbers for viewing, and bookmark pages. Output is automatically displayed in one of three formats—plain text, HTML, and rich text format. You can use the File Viewer shown in Figure A, or an alternate viewer. For information on viewing output with alternate viewers, see topic 4.6 Using File Associations.

Procedure

To view output:

1. Open the File Viewer shown in Figure A in one of two ways:
   - From The Jobs/Explorer window, right-click a job in the History or Backlog and select Output from the pop-up menu. For more information on accessing the File Viewer from the Jobs/Explorer tab, see topic 5.9.5 Viewing Job Output.
   - From The Output window, select a job and click the View button. For more information on accessing the File Viewer from the Output window, see topic 4.2 Working with the Output Window.

2. Scroll through the output by using the horizontal and vertical scroll bars, view a desired page or line number, and/or perform searches by going to the Options menu and selecting Find.
   
   You can cut and paste text from the File Viewer.

Viewing by Page and Line Numbers

You can jump to a specific page and/or line numbers by entering them into the Page/Line fields at the top of the window and clicking the corresponding Go To button or pressing Enter.
Using the Scroll Bars

You can use the horizontal and vertical scroll bars to move around in the File Viewer. As you hold the mouse button down and move the scroll box, File Viewer displays the file’s output. However, the File Viewer can only display output stored in its buffer. If you scroll outside of the buffers range, the LineNo field changes but the display on the screen does not. When you let go of the mouse button, File Viewer will show the correct page’s display.

Finding Specific Text

To find specific text in the output:

1. On the File Viewer window, go to the Options menu and select Find. AppWorx displays the Find window shown in Figure B.

2. Enter the text you want to find and click Find Next.

Check the Match Case check box for case-sensitive searches. AppWorx searches run from the current location to the end of the file.

Bookmarking Pages

As you view a file, you can bookmark pages for printer by selecting the Mark Page button. Your marked page numbers will display in the Pages box located at the bottom of the screen.

Changing Output Formats and Text Size

AppWorx auto selects HTML and rich text format styles based on each file’s extension (.htm, .html, .rtf). Other files default to plain text view. When using plain text style you can adjust the size of the text used in the display by selecting a value from the Size list box at the bottom of the viewer. This changes the size of the text in the viewer, but it does not impact the size of text used when the report is printed. Output can be viewed with an alternate viewer if a file association has been defined. For information on setting file associations, see topic 4.6 Using File Associations.

Viewing the End of Files

To view the end of a text file, go to the Options menu on File Viewer window and select Tail. The tailing option enables operators to observe the most recent end of file every 10 seconds by showing the report as it is being printed to standard output. Use this function, similar to the UNIX tail –[filename] command, when trying to diagnose problems.
4.5 Printing Job Output

To print the output from a job using the File Viewer, go to the File menu and select Print.

After viewing the output, you can print the output by going to the File menu and selecting Print. You can print the output to a local or system printer.

**Printing Output**

To print the output:

1. From the File Viewer window, go to the File menu and select Print (see Figure A).
   
   AppWorx displays the Print window shown in Figure B.

2. Select the options and click OK.
   
   If you have bookmarked one or more pages using the Mark Page button, AppWorx will only print the marked pages.
Printing to an AppWorx System Printer

To print the output to any appropriate system printer defined in AppWorx:

1. From the File Viewer, go to the File menu and select System Print (see Figure A).
   AppWorx displays the Choose a printer window shown in Figure C.
2. Select a printer from the drop-down list and click OK.
   AppWorx prints to the system printer you selected.
   If you have bookmarked one or more pages using the Mark Page button, AppWorx will only print the marked pages.

Previewing a Print Job

To preview a print job:

- Go to the File menu and select Print Preview (see Figure A).
  AppWorx opens the Print Preview window shown in Figure D and displays the output as it will be printed.
4.6 Using File Associations

You can associate different types of files with specific viewers. For example, if a report has a `.doc` extension, you can associate the file with Microsoft Word for Windows. To do this, you must have the type association selected for the file in the File Association window. The file association(s) you create are associated with your user logon on this AppWorx machine.

If you are using a Windows based application, you may choose to save copy of the file from the viewer to your PC.

Adding File Associations

To add an output file association:

1. On the desktop, go to the Options menu and select File Associations.

   AppWorx Opens the File Association window shown in Figure A.

2. Enter a file pattern in the **Pattern** field.

   The **Pattern** field accepts valid UNIX regular expressions (see Regular Expression Examples in this topic). For information on syntax accepted by regular expressions, see [Appendix D: Regular Expression Tables](#).

   **Note:** Regular expressions are case sensitive.

3. In the **Application** field, type the name of the application you wish to associate the file type with, or use the `...` button to browse for it.

4. Select an option from the **Use** field.

<table>
<thead>
<tr>
<th>To:</th>
<th>Select:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always open file type in the associated file viewer.</td>
<td>Always</td>
</tr>
<tr>
<td>To prompts the user before opening in the associated file viewer.</td>
<td>Ask</td>
</tr>
<tr>
<td>To not use the associated file viewer or temporarily disable the association.</td>
<td>Never</td>
</tr>
</tbody>
</table>

5. Click **Add**.

   AppWorx displays the file association in the table at the top of the screen.
UNIX Regular Expression Examples

Example 1. To set an association for any file with ‘ACCOUNTING’ in its name, enter ACCOUNTING in the Pattern field.

Example 2. To set an association for any file names that start with the text ‘ACCOUNTING’, enter ^ACCOUNTING in the Pattern field.

Example 3. To set an association for file names that start with the text ‘ACCOUNTING’, followed by a dot, followed by one or more numbers that include a decimal point and any other characters, and end with .log, enter ^ACCOUNTING[:digit:]+.*\.log$ in the Pattern field. This pattern will match files with names such as ACCOUNTING.1400.2.log.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Matches at the beginning of a line</td>
</tr>
<tr>
<td>ACCOUNTING</td>
<td>Module name</td>
</tr>
<tr>
<td>[:digit:]+.*</td>
<td>Job ID: Numeric characters followed by a decimal point and zero or more characters</td>
</tr>
<tr>
<td>.log$</td>
<td>Dot log extension at the end of the line</td>
</tr>
</tbody>
</table>

For information on syntax accepted by regular expressions, see Appendix D: Regular Expression Tables.

Editing and Deleting File Associations

To edit or delete an output file association:

1. Select a row you wish to edit from the properties table at the top of the File Association window.
   
   AppWorx displays the file type’s attributes in the fields below.

2. Edit the fields.

3. Click Update or Delete.

   AppWorx displays the new file association data in the table at the top of the screen.
5.1 Introduction to Monitoring and Managing Jobs

You can monitor and manage queues, agents, and jobs with the Jobs or Explorer window.

From the Jobs or Explorer window, you can monitor and manage queues, agents, and jobs. You can check the status of queues, change queue settings, and assign queues to thread schedules to control the number of jobs the queue can process. You can check agent status, and start, stop, idle, or resume agents. You can check the status of jobs, restarting or killing them if necessary. Also, you can view and query the History or filter by an agent and/or queue. A sample Explorer screen showing the Backlog and History is shown in Figure A.

The Backlog and History

When a job is submitted to an AppWorx queue, it is sent to the Backlog. Jobs remain in the Backlog until they complete successfully. If a job aborts or is assigned a status other than FINISHED, it will remain in Backlog until an AppWorx operator resets or deletes it.

Before a job executes, you can view and change its parameters by right-clicking on it in the Backlog. When a job completes executing, AppWorx moves it to the History.

You can change the number of jobs and the minutes jobs remain in the History by going to the Options menu on the desktop and selecting Settings. For more information on editing user settings, see topic 2.7 Editing User Settings.

How long a job is retained in the History is determined by the prompt setting for the HISTORY_PURGE module in the SYSTEM chain. Your AppWorx administrator is responsible for setting this value.

Figure A. The Explorer window
Viewing Output and Job Details

You can view output as well as job details for any job in the Backlog or History. You view the job details for a job by right-clicking it and selecting an option from the pop-up menu. For more information, see topic 5.9 Viewing and Editing Job Details.

Viewing the Status Bar

The status bar is displayed across the bottom of the Explorer (or Jobs) window. Its color alerts you to the status of the AppWorx master, agents, and jobs running in the Backlog. When the Explorer (or Jobs) window is minimized, the button on the taskbar uses the same color scheme. For more information, see topic 5.4 Monitoring with the Status Bar and Object Icons.

Using the Explorer or Jobs Windows

The Jobs window can be used to monitor and manage jobs, queues, and agents. It can be upgraded to Explorer, which allows you to focus the display of jobs in the Backlog. The topics that follow describe both windows.

Viewing Chain Summaries and Forecasts

With the Chain Summary and Forecast features, you can view chains and their components in the Backlog and History and view a list of scheduled jobs. For more information, see chapter 6: Viewing Summaries and Forecasts

Using the AppWorx Command Line Functions

You can manage masters, agents, queues, processes, and jobs from the command line. For more information, see chapter 7: Using the AppWorx Command Line Interface.
5.2 Using the Explorer Window

Explorer is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog, and to focus the display of jobs in the Backlog.

Explorer is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog, and to focus the display of jobs in the Backlog. You can also preview and/or print the object tree or the displayed focus of the Backlog (the entire Backlog or jobs by ad hoc, agents, queues, chains, or status).

![Figure A](image.png)

*Figure A. The Explorer window includes the icon tree in the left pane.*

The Explorer window shown in Figure A includes the following three panes:

- The navigation pane on the left side of the screen provides a tree structure with selectable object icons.
- The content pane shown on the top right side of the screen displays the Backlog (jobs waiting to be processed), a summary of objects selected in the object tree, an agent summary, or a queue summary.
- The History pane shown on the bottom right side of the screen displays jobs that have completed executing.

Each of the panes can be expanded by dragging the splitter bars, or clicking the splitter bar arrows.
Explorer Terms

The following Explorer terms are used in this guide:

**Object tree**: The graphical model displayed on the left pane of the Explorer window.

**Object icons**: The icons used in the object tree to represent objects such as jobs, chains, and queues.

**Object keys**: The icons to the left of the expandable objects in the object tree. You can click the object keys to show the objects belonging to the object.

Printing the Object Tree or a Table

You can preview and/or print the object tree or the displayed focus of an Explorer table by selecting the appropriate options from the File menu. Figure B shows the print preview screen for a job in the Backlog that has aborted.

![Print Preview window for a job in the Backlog.](image)

**Figure B.** The Print Preview window for an aborted job in the Backlog.

Upgrading to Explorer

To upgrade to Explorer, call AppWorx Technical Support at 1-877-APPWORX.
5.3 Using the Jobs Window

The Jobs window is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog.

The Jobs window is used to monitor and manage the status of your agents, queues, and jobs running in the Backlog.

The Jobs window shown in Figure A includes two panes:

- The top pane displays the Backlog (jobs waiting to be processed), a summary of agents, or a summary of queues. You can choose the display by selecting the Backlog, Agents, or Queues radio buttons.

- The bottom pane displays the History (jobs that have completed executing).

The panes can be expanded by dragging the splitter bar, or clicking its arrows.

Note: The remaining topics in this guide will be presented showing Explorer enabled.

Upgrading to Explorer

To upgrade to Explorer, call AppWorx Technical Support at 1-877-APPWORX.
5.4 Monitoring with the Status Bar and Object Icons

The status bar is displayed across the bottom of the Explorer (or Jobs) window. Its color alerts you to the status of the AppWorx master, agents, and jobs running in the Backlog. When the Explorer (or Jobs) window is minimized it uses the same color scheme on the taskbar. The object icons for Backlog, Ad Hoc, Agents, Queues, Chains, and Status help you locate aborted/on hold jobs easily.

The status bar shown in Figure A is displayed across the bottom of the Explorer (or Jobs) window. Its color alerts you to the status of the AppWorx master, agents, and jobs running in the Backlog. When the Explorer (or Jobs) window is minimized, the button on the taskbar uses the same color scheme.

The status bar displays the time that the display was last refreshed. For information on editing user settings, see topic 2.7 Editing User Settings.

The virtual workday is displayed to the left of the status bar. It is used to establish reset times for predecessor statements. For information on predecessor statements and setting the virtual workday, see chapter 12: Assigning Predecessor Statements.

The current date and time of the database are displayed to the right of the status bar.

---

### Table: Status Bar Colors and Descriptions

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Green</td>
<td>All jobs, masters, and agents are running satisfactorily.</td>
</tr>
<tr>
<td>Yellow</td>
<td>One or more jobs are on hold. <strong>Note:</strong> If there are aborted jobs and jobs on hold, the aborted jobs take precedence and the status bar will be red.</td>
</tr>
<tr>
<td>Red</td>
<td>One or more jobs have aborted, or otherwise not completed with a status of FINISHED. Or a master or agent has a BUSY or TROUBLE status.</td>
</tr>
</tbody>
</table>

For a description of the master/agent status values, see Appendix A: Master/Agent Status Values. For a description of the job status values, see Appendix B: Job Status Values.
Managing Job Statuses with the Object Icons

With Explorer activated, the icons in the object tree alert you to job status and give you a quick method for finding aborted/on hold jobs.

Use the icons to view:

- The entire Backlog
- Single modules submitted ad hoc
- Expandable chains and their components
- Jobs in the Backlog by status

The following icons are displayed in the Explorer object tree to alert you to the status of jobs in the Backlog and viewable History.

<table>
<thead>
<tr>
<th>Item</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad Hoc</td>
<td><img src="image1" alt="Colored" /></td>
<td>All ad hoc jobs are running satisfactorily, or no ad hoc jobs are running in the Backlog.</td>
</tr>
<tr>
<td></td>
<td><img src="image2" alt="Covered by a yellow triangle" /></td>
<td>One or more jobs are on hold.</td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="Covered by a red X" /></td>
<td>One or more jobs have aborted, or otherwise not complete with a status of FINISHED.</td>
</tr>
<tr>
<td>Chains</td>
<td><img src="image4" alt="Colored" /></td>
<td>All jobs in the chain(s) are running satisfactorily.</td>
</tr>
<tr>
<td></td>
<td><img src="image5" alt="Covered by a yellow triangle" /></td>
<td>One or more jobs are on hold</td>
</tr>
<tr>
<td></td>
<td><img src="image6" alt="Covered by a red X" /></td>
<td>One or more jobs have aborted, or otherwise not complete with a status of FINISHED.</td>
</tr>
<tr>
<td></td>
<td><img src="image7" alt="Gray" /></td>
<td>All jobs in this the chain are complete.</td>
</tr>
<tr>
<td></td>
<td><img src="image8" alt="Outlined" /></td>
<td>The chain has been deleted.</td>
</tr>
<tr>
<td>Jobs</td>
<td><img src="image9" alt="Colored" /></td>
<td>The job is waiting to run. This symbol is used for single modules and chain components.</td>
</tr>
<tr>
<td></td>
<td><img src="image10" alt="Covered by a green dot" /></td>
<td>The job is running satisfactorily.</td>
</tr>
<tr>
<td></td>
<td><img src="image11" alt="Covered by a yellow triangle" /></td>
<td>The job is on hold.</td>
</tr>
<tr>
<td></td>
<td><img src="image12" alt="Covered by a red X" /></td>
<td>The job is currently aborted or killed.</td>
</tr>
<tr>
<td></td>
<td><img src="image13" alt="Covered by a dark red +" /></td>
<td>A record written to the History of a job that has aborted or been killed. This icon is only seen when the job is part of a chain.</td>
</tr>
<tr>
<td></td>
<td><img src="image14" alt="Gray" /></td>
<td>The job has completed.</td>
</tr>
<tr>
<td></td>
<td><img src="image15" alt="Outlined" /></td>
<td>The job has been deleted.</td>
</tr>
</tbody>
</table>
5.5 Example: Finding an Aborted Job in a Chain

Below is an example of how a user can find and resubmit an aborted job in a chain with Explorer.

In this example, CHAIN_1 has been called. CHAIN_1 includes seven components: two modules, and five chains. The first three icons represent chains that ran successfully. When the fourth chain ran, a job aborted, leading to the sequence of events below.

1. When the job aborted, the status bar turned red to alert the user Pat Brown.
2. Pat opened the Explorer window and could see by the red X on the Chains icon that one or more jobs in a chain had aborted.
   
   Note: Pat could have selected the Backlog or Status icons to find the aborted job, but chose the chain option out of personal preference.

3. Next, Pat clicked the Chains icon key to view a list of running chains. The icon for CHAIN_1 had a red X, so Pat knew that this chain, or a chain nested therein, contained an aborted job.

4. Pat clicked the CHAIN_1 icon key to view the list of jobs belonging to CHAIN_1.
   Pat could see that all of the jobs in the first three chains had finished successfully because the chain icons were gray, and that a job had aborted in the fourth chain (because the chain icon had a red X).

5. Pat clicked the BUSINESS_REPORTS icon key to view the components in that chain.
   Pat could see by the red X on the forth module’s icon that it had aborted. A second listing for the job, a gray icon with a dark red + (notice that it also lists the 004 job number to alert the user that it represents the same job) notes that a record of the job aborting was written into the HISTORY.

Now Pat could view the job details for the job to see why it aborted, and reset or delete it.
5.6 Viewing Jobs in the Backlog and History

Clicking the Backlog tree object icon displays jobs waiting to be processed.

When a job is submitted to an AppWorx queue, it is sent to the Backlog. Jobs remain in the Backlog until they complete successfully. If a job aborts, or is assigned a status other than FINISHED, the job will remain in Backlog until an AppWorx operator resets or deletes it.

Before a job executes, you can view and change its parameters by right-clicking on it in the Backlog. When a job completes executing, AppWorx moves it to the History.

You can change the number of jobs or minutes they remain in the History by going to the Options menu and selecting Settings. For more information, see topic 2.7 Editing User Settings.

How long a job is archived is determined by the prompt setting for the HISTORY_PURGE module in the SYSTEM chain. Your AppWorx administrator is responsible for setting this value.

To view all jobs in the Backlog, click the Backlog icon in the object tree shown in Figure A. To search for jobs in the Backlog, you can focus the view of jobs by selecting an icon from the object tree in Explorer. For more information, see topic 5.7 Focusing the Backlog Display with Explorer.

The History is displayed in the lower pane of the screen. You can run a query to search for jobs in the History. For more on running a query of the History, see topic 5.14 Querying for Jobs in the History.
Column Descriptions for the Backlog and History

The columns in the Backlog and History are described below. Columns unique to either the Backlog or History are noted.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>Name of the queue.</td>
</tr>
<tr>
<td>Module</td>
<td>The name, or alias if defined, of the job.</td>
</tr>
<tr>
<td>Job Started</td>
<td>Displayed in the Backlog. The date and time the job is scheduled to start executing. After a job begins executing, the date and time it started.</td>
</tr>
<tr>
<td>Job Finished</td>
<td>Displayed in the History. The date and time the job finished executing.</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>Displayed in the History. The elapsed time the job ran.</td>
</tr>
<tr>
<td>Status</td>
<td>The current status of the job. For a list of status values, see Appendix B: Job Status Values.</td>
</tr>
<tr>
<td>Agent</td>
<td>The agent or agent group where the job will execute (Backlog), or was executed (History).</td>
</tr>
<tr>
<td>Requestor</td>
<td>The person that submitted the job or the user entered as the requestor of the schedule. If nothing is displayed, a default requestor was not specified.</td>
</tr>
<tr>
<td>Chain</td>
<td>The chain that contained the component. If the job is not part of a chain, this field will be blank.</td>
</tr>
<tr>
<td>Job ID</td>
<td>The identification number, which AppWorx auto-assigns to the job. Note: Decimal values at the end of Job IDs indicate that a job was restarted. A job has .01 added to its Job ID each time it is restarted.</td>
</tr>
<tr>
<td>Log</td>
<td>Uses a Y to identify jobs that includes operator logs.</td>
</tr>
</tbody>
</table>
5.7 Focusing the Backlog Display with Explorer

If you have the Explorer add-on product, you can select an icon from the tree to limit the jobs listed in the Backlog.

Using Explorer, you can select an icon to limit the jobs listed in the Backlog. An example Explorer screen is shown in Figure A.

The icon you select in the object tree determines the jobs listed in the Backlog.

<table>
<thead>
<tr>
<th>To list:</th>
<th>Select this icon:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All jobs in the Backlog.</td>
<td>Backlog</td>
</tr>
<tr>
<td>Jobs that have been submitted using the Requests window.</td>
<td>Ad Hoc</td>
</tr>
<tr>
<td>Jobs that are set to run on a particular agent, or agent group.</td>
<td>Agents</td>
</tr>
<tr>
<td>Jobs that are set to run on a particular queue.</td>
<td>Queues</td>
</tr>
<tr>
<td>Jobs that are part of a particular chain.</td>
<td>Chains</td>
</tr>
<tr>
<td>Jobs with a particular job status (WAITING, RUNNING, ABORTED, or HOLD).</td>
<td>Status</td>
</tr>
</tbody>
</table>

**Note:** You can also filter the display of agents and/or queues by going to the Filter menu and selecting Query. For more information, see topic 5.15 Filtering the Backlog and History by Agents and Queues.
Procedure

To limit the view of jobs in the Backlog by object:

1. If necessary, click the icon’s key to display the child objects. To view jobs in a chain, you may need to open several icon keys (see Figure B).

   ![Figure B. Click icon keys to expand a icon.](image)

   **Note:** Chain components display an object icon, their order in the chain, an (S) or (M) for single or multi-threaded, and their name/alias.

2. Select an object from the list of icons.

   AppWorx displays the list of jobs belonging to that object in the upper right pane (see Figure C).

   ![Figure C. Select an icon to view the jobs belonging to it.](image)
5.8 Changing Job Statuses in the Backlog

The Backlog shows the jobs being processed in each queue. You can change the status of a job and delete jobs from the Backlog by right-clicking and selecting a status option.

You can select one or more jobs in the Backlog (shown in Figure A) and right-click to:

- Put jobs on hold
- Kill jobs
- Resubmit aborted, killed, or on hold jobs
- Delete jobs

Putting Jobs On Hold

If a job is in the Backlog but has not yet started, you can put it on hold. The job will remain in the Backlog with a status of HOLD until you reset it or delete it from the queue. If a job has started running, you cannot put it on hold. However, you can kill a running job.

Killing Jobs

If a job is running, you can kill it by selecting the job and using the Kill command. When you kill a job, it stays in the Backlog until you delete it, or reset it. When you kills a job, AppWorx makes an entry in the History showing the job was killed.
Resubmitting Aborted, Killed and On Hold Jobs

If a job aborts and remains in the Backlog, is killed, or is put on hold, you can resubmit it directly from the Job/Explorer window. Before restarting a job, you can review its parameters, prompts, and conditions, and correct any problems. When you restart an aborted job, its status changes to NULL. As soon as a thread becomes available in the queue, the status changes to STARTING.

An aborted job stays in the Backlog if the Stay in queue on abort option was set when the module was created. If this option was not set, the job is cleared from the Backlog and an entry is displayed in the History. You cannot restart a job from the Jobs/Explorer window once it has been removed from the Backlog. However, you can resubmit the job by going to the Requests window.

Deleting Jobs

If a job in the Backlog has a status other than STARTING or RUNNING, you can delete it. For example, jobs with a status of THREAD WAIT, ABORTED or KILLED can be deleted. When you have deleted a job, you cannot resubmit it directly from the Jobs/Explorer window.

Subtopics

Additional information is provided in the following subtopics:

5.8.1 Putting Jobs On Hold
5.8.2 Killing Jobs
5.8.3 Resubmitting Aborted and On Hold Jobs
5.8.4 Deleting Jobs
5.8 Changing Job Statuses in the Backlog

5.8.1 Putting Jobs On Hold

Before a job runs, you can put it on hold. To put one or more jobs on hold, select the job(s) and right-click. Choose the Hold option from the pop-up menu.

If a job is in the Backlog but has not yet started, you can put it on hold. The job will remain in the Backlog with a status of HOLD until you reset it or delete it from the queue. If a job has started running, you cannot put it on hold. However, you can kill a running job. For more information on killing a job, see topic 5.8.2 Killing Jobs.

Procedure

To put jobs on hold:

1. Select one or more jobs in the Backlog and right-click.
   
   AppWorx displays the pop-up menu shown in Figure A. The number to the right of the Hold, Kill, Reset and Delete options references the number of jobs you have selected that are eligible for that operation.

2. Select the Hold option.
   
   AppWorx will display the Job Hold confirmation window.

3. Click Yes to hold the job(s).
   
   AppWorx closes the window and changes the jobs’ status to HOLD.
**Result**

When you put jobs on hold, their status is displayed as HOLD. AppWorx displays the status bar at the bottom of the window in yellow to alert you that one or more jobs are on hold (see Figure B). The job remains on hold until you reset the status or delete the job from the Backlog.

**Taking Jobs Off Hold**

To take jobs off hold:

1. Select one or more jobs with a HOLD status in the Backlog and right-click.
2. Click the Reset option.
   
   AppWorx will display the Job Reset confirmation window.
3. Click Yes to reset the job(s).

   AppWorx closes the window and changes the job's status to NULL, then to STARTING as soon as there is a thread available in the queue assigned to the job.

---

**Figure B.** A yellow status bar signifies that one or more jobs are on hold.
If a job is running, you can kill it from the Backlog by changing its status to KILL. To kill one or more jobs, select the job(s), and right-click. Choose the Kill option from the pop-up menu.

If a job is running, you can kill it by selecting the job and using the Kill command. When you kill a job, it stays in the Backlog until you delete it, or reset it. When you kills a job, AppWorx makes an entry in the History showing the job was killed.

**Procedure**

To kill running jobs:

1. Select one or more running jobs in the Backlog that you wish to kill and right-click. AppWorx displays the pop-up menu shown in Figure A. The number to the right of the Hold, Kill, Reset and Delete options references the number of jobs you have selected that are eligible for that operation.

2. Click the Kill option. AppWorx will display the Job Kill confirmation window.

3. Click **Yes** to kill the job(s). AppWorx closes the window and kills the jobs. AppWorx changes the status of the job(s) to KILLED and makes an entry in the History showing the jobs that were killed.
Result

When you kill a job from the Backlog, AppWorx issues the `kill -15` UNIX command followed by the `kill -9` UNIX command. The process is stopped, the status of the job in the Backlog changes to KILLED, and an entry is made in the History showing the job was killed (see Figure B). The killed job stays in the Backlog until you delete it or reset it.

*Figure B.* When you kill a job the process is stopped, the status of the job in the Backlog changes to KILLED, and an entry is made in the History showing the job was killed.
5.8 Changing Job Statuses in the Backlog

5.8.3 Resubmitting Aborted and On Hold Jobs

If a job aborts or is put on hold, you can resubmit it directly from the Backlog. To resubmit one or more jobs, select the job(s), and right-click. Choose the Reset option from the pop-up menu.

If a job aborts and remains in the Backlog, is killed, or is put on hold, you can resubmit it directly from the Job/Explorer window. Before restarting a job, you can review its parameters, prompts, and conditions, and correct any problems. When you restart an aborted job, its status changes to NULL. As soon as a thread becomes available in the queue, the status changes to STARTING.

An aborted job stays in the Backlog if the Stay in queue on abort option was set when the module was created. If this option was not set, the job is cleared from the Backlog and an entry is displayed in the History. You cannot restart a job from the Jobs/Explorer window once it has been removed from the Backlog. However, you can resubmit the job by going to the Requests window.

Figure A. You can resubmit one or more aborted or on hold jobs.
Procedure

To resubmit one or more jobs from the Backlog:

1. If appropriate, change the job details.
   For more information on changing job details, see topic 5.9 Viewing and Editing Job Details.

2. Select one or more jobs in the Backlog and right-click.
   AppWorx displays the pop-up menu shown in Figure A. The number to the right of the Hold, Kill, Reset and Delete options references the number of jobs you have selected that are eligible for that operation.

3. Select the Reset option.
   AppWorx will display the Job Reset confirmation window.

4. Click Yes to reset the job(s).

Result

AppWorx closes the window and changes the status of the job(s) to NULL (see Figure B). When a thread becomes available for the queue, AppWorx will start the job(s).

*Figure B. Resubmitted jobs go to a NULL status. When a thread becomes available for the queue, AppWorx starts them.*
Chapter 5: Monitoring and Managing Jobs

5.8 Changing Job Statuses in the Backlog

5.8.4 Deleting Jobs

If a job has a status other than STARTING or RUNNING, you can delete it from the queue. To delete one or more jobs, select the job(s), and right-click. Choose the Delete option from the pop-up menu.

If a job in the Backlog has a status other than STARTING or RUNNING, you can delete it. For example, jobs with a status of THREAD WAIT, ABORTED or KILLED can be deleted. When you have deleted a job, you cannot resubmit it directly from the Jobs/Explorer window.

**Procedure**

To delete jobs:

1. Select one or more jobs in the Backlog and right-click.
   - AppWorx displays the pop-up menu shown in Figure A. The number to the right of the Hold, Kill, Reset and Delete options references the number of jobs you have selected that are eligible for that operation.

2. Select the Delete option.
   - AppWorx will display the Job Delete confirmation window.

3. Click Yes to delete the job(s).
   - AppWorx closes the window and deletes the jobs.
Result

When you delete jobs, they are removed from the Backlog. A record of the job deleted from the Backlog is now displayed in the History with a **DELETED** status (see Figure B). After deleting a job, you cannot resubmit it from the Jobs/Explorer window.

![Figure B](image.png)

*Figure B.* Jobs deleted from the Backlog are displayed in the History with a **DELETED** status.
5.9 Viewing and Editing Job Details

Before jobs run, or when they abort or are killed, you can change their parameters in the Backlog. To change a job's parameters, right-click the job and choose Job Details from the pop-up menu. When you change a module or chain's parameters in the Backlog the changes are applied only to that instance of the job—they do not affect the definition of the module/chain.

You can right-click any job in the Backlog or History and select Job Details from the pop-up menu to view the job's General job details (see Figure A).

Figure A. Right-click Job Details to view the parameters for a job.

**Note:** If you have a large number of output files or jobs in the Backlog, AppWorx will display a message alerting you that the display has been truncated.

Before a job runs, or when a job aborts, you can change the parameters of the job from the General tab on the job details window shown in Figure B. You can change the queue, printer, output option, number of copies, start date, login, and agent. You can also add, delete, or change the prompt values and delete conditions for the job. The changes you make apply only to the current instance of the job, and will not affect the job the next time it is submitted.

Job output can be viewed from the History, and the notes for a job can be viewed in the History or Backlog.

**Note:** To edit the parameters for jobs submitted by other users, you must have the All User Backlog Edits user option assigned to you by your AppWorx administrator. For more information on setting user options, see topic 16.3 Setting User Options.
Viewing and Editing General Job Details

To change the parameters associated with a job in the Backlog:

1. Right-click the job and select Job Details from the pop-up menu (see Figure A). AppWorx displays the job details window with the General tab selected (see Figure B).

2. Make the desired changes on the General tab.

3. You can select the other tabs to view and/or edit the prompts, predecessors, conditions, notes, and output for the job.

4. To accept the changes on all tabs, click OK.

Subtopics

The following subtopics detail viewing and/or editing of the job details tabs.

5.9.1 Viewing and Editing Job Prompts
5.9.2 Viewing Job Predecessors
5.9.3 Viewing Job Conditions
5.9.4 Viewing Job Notes
5.9.5 Viewing Job Output
5.9 Viewing and Editing Job Details

5.9.1 Viewing and Editing Job Prompts

To view the prompts of a job in the Backlog or History, right-click the job and choose the Prompts option from the pop-up menu. You can edit the prompts for jobs in the Backlog. When you edit prompts for a module/chain in the Backlog the changes are applied only to that instance of the job—they do not affect the definition of the module/chain.

Prompts pass user input to the program run by a job. Prompts for a job can be viewed and edited from the Backlog and viewed from the History.

**Viewing Prompts**

To view the prompts for a job, right-click a job in the Backlog or History and choose Prompts (see Figure A). AppWorx displays the Prompts window shown in Figure B.

![Figure A. To view prompts select a job and right-click.](image)

![Figure B. The Prompts tab](image)
**Editing Prompts in the Backlog**

To edit the prompts for a job in the Backlog:

1. Right-click the job and choose Prompts from the pop-up menu (see Figure A). AppWorx displays the job details window with the Prompts tab selected (see Figure B).
2. Click to select a prompt.
3. You can edit the prompt values for each job in the Backlog in the following ways:
   - Edit the data in the Value column and press Enter to change the prompt value.
   - To set the prompt value to the default value click Default.
   - To select a value from a list of values (if available) click LOV.
   - To select a value from a multi select list of values (if available) click MS.

When you edit prompts for a module/chain in the Backlog, the changes are applied only to that instance of the job—they do not affect the definition of the module/chain. You can select the other tabs to view and/or edit the general job details, predecessors, conditions, notes, and output for the job.

4. Click OK to accept the changes and exit to the Jobs/Explorer window.
5.9 Viewing and Editing Job Details

5.9.2 Viewing Job Predecessors

To view the predecessor statements of a job in the Backlog or History, right-click the job and choose the Predecessors option from the pop-up menu. To add, edit, or delete predecessors in the Backlog select the appropriate button on the Predecessors tab. When you add, edit, or delete predecessors for a module/chain in the Backlog the changes are applied only to that instance of the job—they do not affect the definition of the module/chain.

Predecessor statements must be met before a job will be eligible to run. They are evaluated prior to any BEFORE conditions the job might have. They can be viewed from the Backlog and History.

**Procedure**

To view the predecessor statements for a job, right-click the job and select Predecessors from the pop-up menu (see Figure A). AppWorx displays the job details window with the Predecessors tab selected (see Figure B).

You can add, edit, or delete predecessors for jobs in the Backlog. When you add, edit, or delete predecessors for a module/chain in the Backlog the changes are applied only to that instance of the job—they do not affect the definition of the module/chain. For more information, see topic 12.3 Adding Predecessor Statements to Modules and Chains. You can select the other tabs to view and/or edit the general job details, prompts, notes, and output for the job.

Figure A. Right-click to view the predecessors for jobs in the Backlog or History.
**Figure B.** The Predecessors tab displays the predecessors for a job.
5.9 Viewing and Editing Job Details

5.9.3 Viewing Job Conditions

To view the conditions of a job in the Backlog or History, right-click the job and choose the Conditions option from the pop-up menu. To add, edit, or delete conditions in the Backlog select the appropriate button. When you add, edit, or delete conditions for a module/chain in the Backlog, the changes are applied only to that instance of the job—they do not affect the definition of the module/chain.

Conditions control the execution of jobs. They can be evaluated before, during, and after a job executes, or after a job is deleted. Conditions for a job can be viewed from the Backlog and History. Conditions can be added, edited, or deleted from jobs in the Backlog.

Procedure

To view the conditions for a job, right-click the job and select Conditions from the pop-up menu (see Figure A). AppWorx displays the job details window with the Conditions tab selected (see Figure B).

To add, edit, or delete a condition for a job in the Backlog, select the appropriate button. When you add, edit, or delete conditions for a module/chain in the Backlog, the changes are applied only to that instance of the job—they do not affect the definition of the module/chain. For more information on defining conditions, see chapter 13: Working with Conditions. You can select the other tabs to view and/or edit the general job details, prompts, notes, and output for the job.
Column Descriptions

The following information will be listed for each condition.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing</td>
<td>The timing for the condition. Conditions can be set to run before (B), during (D), or after (A) the job executes, or when the job has been deleted (R).</td>
</tr>
<tr>
<td>Performed</td>
<td>Displays the timing for the condition’s action. There are three possible values for this column:</td>
</tr>
<tr>
<td></td>
<td>• Once: <strong>The first time the condition is true</strong> is selected as a condition’s action timing, and it has not yet taken the action.</td>
</tr>
<tr>
<td></td>
<td>• Continuous: <strong>Every time the condition is true</strong> is selected as a condition’s action timing; it may have taken the action one or more times.</td>
</tr>
<tr>
<td></td>
<td>• Done: <strong>Disabled</strong> is selected as a condition’s action timing. Conditions are done when they have <strong>The first time the condition is true</strong> set as their action timing and they have taken the action. You can also select <strong>Disabled</strong> in a module or chain’s definition if you want to inactivate the condition.</td>
</tr>
<tr>
<td>Condition #1</td>
<td>The first condition statement.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>The qualifier for Condition #1 and Condition #2.</td>
</tr>
<tr>
<td>Condition #2</td>
<td>The second condition statement.</td>
</tr>
<tr>
<td>Action</td>
<td>The action of the condition.</td>
</tr>
<tr>
<td>Arguments</td>
<td>The arguments for the action. Some of the actions require that you specify one or more arguments.</td>
</tr>
</tbody>
</table>
5.9  Viewing and Editing Job Details

5.9.4  Viewing Job Notes

To view the notes for a job in the Backlog or History, right-click the job and choose the Notes option from the pop-up menu. If Notes is grayed out on the pop-up menu, the job has no notes.

Notes are written by the person who created the module or chain, they provide relevant information about the processing of a job. You can access these comments, suggestions, or instructions from the Backlog or History when they have been included.

Three types of notes may be included to list information specific to a given job.

- Module notes are used when a module is going to be requested ad hoc, or when the notes for a module would be useful—regardless of how it is invoked.
- Chain notes are used to provide information about the entire chain.
- Chain detail notes are used to provide information specific to a chain component.

Each type of note may include an abort and a general category. Abort notes provide instruction or information for when the job aborts. They are displayed regardless of whether the job has actually aborted.
Procedure

To view the notes for a job in the Backlog or History:

1. Right-click the job in the Backlog or History and select Notes from the pop-up menu. If Notes is grayed out on the pop-up menu, the job has no notes.
   
   **Note:** You can right-click chain components in the Backlog or History panes or from the icon tree as shown in Figure A.

   AppWorx displays the job details window with the Notes tab selected (see Figure B).

   ![Figure B. The Notes tab of the job details window.](image)

2. In the top portion of the window, select the type of notes you wish to view.

   AppWorx displays the notes you selected in the bottom portion of the window.

   You can select the other tabs to view and/or edit the general job details, prompts, predecessors, conditions, and output for the job.

3. Click **OK** to close the window.
5.9 Viewing and Editing Job Details

5.9.5 Viewing Job Output

To view the output of a job in the History, right-click the job and choose the Output option from the pop-up menu. Click View in the Job Output window to view the job’s output with File Viewer.

After a job completes executing, it is moved into the History. You can see how the job was executed by viewing the job’s log file, and its output.

Procedure

To view the output generated by a job after it completes executing:

1. Right-click the job and choose Output from the pop-up menu (see Figure A).
   AppWorx lists the system and output files for the job on the Output tab of the job details window. Standard output and error file names begin with an ‘o’. Body output files begin with a ‘b’ (see Figure B).

2. To view a file, select the file and click View.
   AppWorx displays the File Viewer. For information on using the File Viewer and printing output, see chapter 4: Viewing and Printing Output.
You can select the other tabs to view and/or edit the general job details, prompts, predecessors, conditions, notes for the job.
5.10 Working with Operator Logs

To view or add an operator log for a job in the Backlog or History, right-click the job and choose the Operator log option from the pop-up menu.

You can include an operator log to provide relevant information about the processing of a job. You can access, add, and query logs from the Backlog or History.

Viewing and Adding Logs for a Job

To add an operator log to a job:

1. Right-click the job and choose Output from the pop-up menu (see Figure A).
   AppWorx opens the Operator log window with the Current log tab selected (see Figure B). If there are any entries for the job, they are displayed in the Log box. Each entry includes the user name of the person who wrote the log, and the date and time it was submitted. In Figure B, there is one entry associated with this job.

2. To add an entry to the job, enter text in the New Entry box and click Add.
   AppWorx adds the entry to the Log box.

3. Click OK to save the entry and close the Operator log window.
Viewing Other Logs

You can view logs for other instances of this or any other jobs. To view other logs:

1. From the Operator log window, select the Other logs tab.
   AppWorx displays all previous logs for the module/chain in the table on the top of the screen as shown in Figure C.

2. Select a log from the table to view it’s text below.
3. To view logs for other modules/chains, select the module/chain from the Module drop-down box. You can also query by keywords in the text.
5.11 Monitoring Agents

You can view an Agent Summary by clicking the Agents icon from the object tree. You can also view an agent’s log by right-clicking it.

In AppWorx, each instance of AppWorx is referred to as an agent. The agents can be on the same machine, or on different machines. For more information on agents, see chapter 17: Working with Agents.

Each master has one local agent and can control numerous remote agents. Two remote agents (PROD2 and SAP1) have been defined in Figure A.

Viewing the Agent Summary

To view an Agent Summary, select the Agents icon. AppWorx displays the agent summary in the top right pane of the screen (see Figure A).

Viewing Agent Logs

To view the process logs for a master or agent, right-click on an agent and select View Log from the pop-up menu. The log will display in the File Viewer. For detailed information using the File Viewer, see topic 4.4 Viewing Job Output with the File Viewer. For information on master/agent processes, see chapter 2: AppWorx Processes in the Technical Reference Guide.
Column Descriptions for the Agent Summary

The Agent Summary displays the status of each agent. The columns are described below.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Displays the name of the agent, agent group, or master.</td>
</tr>
<tr>
<td>Status</td>
<td>Displays the current status of the agent. For information on possible status values, see Appendix A: Master/Agent Status Values.</td>
</tr>
<tr>
<td>M/A/G</td>
<td>Identifies the agent as a master (M), agent (A) or group (G). For information on agents, see chapter 17: Working with Agents. <strong>Note:</strong> Although the master and its local agent are defined as a single object in AppWorx, they are listed separately in the Agent Summary, and separate actions can be taken on them.</td>
</tr>
<tr>
<td>Updated</td>
<td>Gives the date and time when the agent data was last updated.</td>
</tr>
<tr>
<td>Max Jobs</td>
<td>Specifies the number of threads available for the agent.</td>
</tr>
<tr>
<td>Bklg</td>
<td>Specifies the number of jobs in the Backlog.</td>
</tr>
<tr>
<td>Run</td>
<td>Specifies the number of jobs RUNNING.</td>
</tr>
<tr>
<td>Hold</td>
<td>Specifies the number of jobs on HOLD.</td>
</tr>
<tr>
<td>Abtd</td>
<td>Specifies the number of jobs ABORTED in the Backlog.</td>
</tr>
<tr>
<td>Chains</td>
<td>Specifies the number of chains in the Backlog.</td>
</tr>
</tbody>
</table>

Descriptions of Icons

The following icons are displayed in the Explorer object tree to alert you to the status of the master and agents.

<table>
<thead>
<tr>
<th>Item</th>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td><img src="image" alt="Gold" /></td>
<td>Gold: The master is running.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Gray" /></td>
<td>Gray: The master is stopped</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Gray with red X" /></td>
<td>Gray with a red X: The master is in a busy or trouble state.</td>
</tr>
<tr>
<td>Agent</td>
<td><img src="image" alt="Gold" /></td>
<td>Gold: The agent is running.</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Gray" /></td>
<td>Gray: The agent is stopped</td>
</tr>
<tr>
<td></td>
<td><img src="image" alt="Gray with red X" /></td>
<td>Gray with a red X: The agent is in a busy or trouble state.</td>
</tr>
<tr>
<td>Group</td>
<td><img src="image" alt="Agent group" /></td>
<td>Agent group icons do not change color based on status.</td>
</tr>
</tbody>
</table>
5.12 Managing Agents

You can start, stop, idle, resume, or reset agents. You can also edit an agent’s threads by right-clicking it.

You manage agents in several ways. From the Explorer window you can:

- Take an action on a single agent by right-clicking an agent’s icon in the object tree or by right-clicking an agent’s listing in the Agent Summary.
- Take an action on one or more agents by highlighting them in the Agent Summary and right-clicking.
- Take an action on all agents by right-clicking the Agents icon in the object tree.
- Change an agent’s thread schedule by right-clicking the agent.

Note: To manage agents, you must have the Manage all Agents user option assigned to you by your AppWorx administrator. For more information on user options, see topic 16.3 Setting User Options.

Creating Agents

Agents are created using the Agents window. For detailed information, see topic 17.2 Defining Agents.
Starting or Stopping Agents

To start or stop agents, select one or more agents, right-click, and choose the Start or Stop option from the pop-up menu. These options are equivalent to issuing Startso and Stopso commands.

Note: It may take some time to update the status when starting and stopping agents.

Idling or Resuming Agents

If you do not want to stop one or more AppWorx agents, but you want to stop processing jobs through the AppWorx queues, select the agent(s), right-click, and choose the Idle option from the pop-up menu. Jobs in the Backlog set to run on an idled agent will have a status of UNAVAILABLE. When the agent is once again available, the jobs will be run. The icons for idled agents will be displayed in yellow in the object tree. To take the agent out of the idle state, select the Resume option.

Taking Agents Offline

When one or more agents are in a BUSY or TROUBLE status (the icons for these agents will be covered by a red X in the object tree), you can right-click them and choose Reset from the pop-up menu. Resetting an agent does not take an action on an agent, it only changes it’s viewable status to STOPPED. This displays a gray agent icon which will not affect the status bar.

Changing Thread Schedules for the Master and its Agents

Master and agent thread schedules set the maximum number of concurrent jobs that can run through any combination of queues. To change a master or agent’s thread schedule, right-click the agent icon and select Threads from the pop-up menu. AppWorx displays the Threads window shown in Figure B where you select a thread schedule. Thread schedules are assigned to masters, agents, and queues. You can also specify a local or remote agent’s thread schedule in its definition. For more on assigning thread schedules to agents, see topic 17.2 Defining Agents.

Figure B. Right-click an agent (or the master) to select its thread schedule.
5.13 Monitoring and Managing Queues

You can control the flow of jobs submitted to your system by assigning different thread schedules and priorities to your queues. You can also inactivate one or more queues.

Controlling the load on your system is critical. In AppWorx, you control the workload by setting the number of concurrent jobs that can pass through a queue.

All jobs submitted to your system from AppWorx must pass through an AppWorx queue. You can control the number of jobs that flow through a queue in several ways:

- Select a thread schedule for the queue based on its **Min thread** and **Max thread** values.

  Minimum threads ensure that you always have a specified number of ‘standby’ threads available for priority ‘rush’ jobs. Maximum threads control the maximum number of jobs that can run concurrently in the queue. For more information on thread schedules, see topic 18.3 Defining Thread Schedules.

- Set a priority for each queue.

- Use the single and multiple thread settings for modules and chains.

Creating Queues

Queues are created using the Queues window. For detailed information, see chapter 18: Administering Queues.
**Viewing the Queue Summary**

To view a Queue Summary, select the Queues icon. AppWorx displays a list of queues in the upper right pane (see Figure A).

**Column Descriptions**

The queue summary displays the status of each queue. The columns are described below.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>Name of the queue.</td>
</tr>
<tr>
<td>Status</td>
<td>Displays whether the queue is active or inactive. Active queues apply the thread settings. When you inactivate a queue, no jobs will be processed through it. The jobs will remain in the Backlog with a NULL status until the queue is activated. <strong>Note:</strong> The object tree icons for inactive queues will be covered by a yellow triangle.</td>
</tr>
<tr>
<td>Pri</td>
<td>The priority assigned to the queue. Priority defines the order in which the queue is scanned for job initiation. Jobs in a higher priority queue will be processed before jobs in a lower priority queue. However, if all threads in a high priority queue are being used, AppWorx will process jobs in lower priority queues until the maximum number of threads is reached. You can control the load on your system and the availability of computing resources by giving careful thought to how you prioritize queues and set their thread limits. The lower the number you assign to a queue, the higher priority it will have to run jobs. For information on setting queue priorities, see topic 18.2 Defining Queues.</td>
</tr>
<tr>
<td>Threads</td>
<td>Number of threads assigned to the thread schedule displayed in the Schedule column.</td>
</tr>
<tr>
<td>Bklg</td>
<td>The number of jobs in the Backlog waiting to run on the queue.</td>
</tr>
<tr>
<td>Schedule</td>
<td>The thread schedule assigned to a queue. Thread schedules control the maximum number of concurrent jobs that can run through the queue at any given time. For information on defining queues and thread schedule, see chapter 18: Administering Queues.</td>
</tr>
</tbody>
</table>

**Changing Queue Settings**

To change a queue’s settings, select the queue, right-click, and select Change from the pop-up menu (see Figure A). You can also activate/inactivate one or more queues in the Queue Summary by highlighting the queues, right-clicking, and selecting the appropriate option.
5.14 Querying for Jobs in the History

To search for specific jobs in the History, go to the Filter menu and select History Query. You can query by queues, modules, chains, job statuses, agents, requestors, start times, and job IDs.

If the number of jobs listed in the History is overwhelming, or you wish to view output from jobs which exceed the settings for the Job History Limits, you can run a history query.

**Procedure**

To search for specific jobs:

1. Go to the Filter menu and select History Query (see Figure A).
   
   AppWorx displays the Query Definition window.

2. Select items from one or more of the labeled boxes/fields and click **OK**.
   
   The boxes/fields are described in Table A.

   Use Shift+Click and Ctrl+Click to select or deselect more than one entry in a list.

   You can click **Clear** at any time to clear all fields.

   To cancel search criteria and close the window, click **Cancel** in the Output Query window.

---

**Figure A.** You can limit the jobs listed in the History using the History Query window.
Table A. History query options

<table>
<thead>
<tr>
<th>Query option</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent/Agent group</td>
<td>In addition to querying, you can view jobs by agent using the Filter option. For more information on the Filter option, see topic 5.15 Filtering the Backlog and History by Agents and Queues.</td>
</tr>
<tr>
<td>Queue</td>
<td>In addition to querying, you can view jobs by queue using the Filter option. For more information on the Filter option, see topic 5.15 Filtering the Backlog and History by Agents and Queues.</td>
</tr>
<tr>
<td>Module</td>
<td>You can use the '_' wildcard to represent a single character, and the '%' wildcard to represent an unlimited number of characters. You can type the first few letters of the module's name to search for it. <strong>Note:</strong> To query for a job that ran under an <strong>Alias</strong>, enter the alias name in this field. You must type in the name because it will not be displayed in the list.</td>
</tr>
<tr>
<td>Status</td>
<td>For a description of the job status values, see Appendix B: Job Status Values.</td>
</tr>
<tr>
<td>Requestor</td>
<td>This list will contain only your user name unless you have been assigned the <strong>View all User Outputs</strong> user option by your AppWorx administrator</td>
</tr>
<tr>
<td>Chain</td>
<td>This option searches for job output for chain components. The chains themselves are not returned by queries, because they produce no output.</td>
</tr>
<tr>
<td>From start time</td>
<td>Set the date/time (if desired) by clicking the desired field. AppWorx displays the Date Time Selector window shown in Figure A.</td>
</tr>
<tr>
<td>To start time</td>
<td></td>
</tr>
<tr>
<td>JobID</td>
<td>When using this field, include the decimal value where appropriate. Decimal values at the end of Job IDs indicate that jobs have been restarted. Every job has .01 added to its JobID each time it is restarted.</td>
</tr>
</tbody>
</table>

Viewing Query Results

AppWorx runs the search and displays the queried results. The **Apply Query** check box will be selected (see Figure B). To view the unqueried History, uncheck the **Apply Query** box. You can view the queried results again by rechecking the **Apply Query** box. If you select the History Query menu item again, AppWorx returns you to the Query Definition window. AppWorx displays the search criteria you defined on your last query. To run a new query, select new data and click **OK**.

![Figure B. The Apply Query check box.](URL)
5.15 Filtering the Backlog and History by Agents and Queues

You can filter the display by agents and or queues.

There may be times when you only want to view jobs running on a particular agent or through a particular queue. You can do this by going to the Filter menu and selecting Filter Backlog and History.

**Procedure**

1. Go to the Filter menu and select Filter Backlog and History (see Figure A).
   AppWorx displays the Select filters window shown in Figure B.
2. Select an agent and/or a queue and click OK.
   AppWorx displays the filtered Backlog and History. The word ‘filtered’ is displayed in the Backlog title bar, and the Filter box is checked in the History title bar (see Figure C).

   **Note:** The filter option will be overridden in the Backlog if you select a focus with an icon. The filter option will be overridden in the History if you apply a History query.
Disabling Filters

To view the unfiltered Backlog and History, uncheck the Filter check box (see Figure C). You can view the filtered results again by rechecking the Filter check box.
5.16 How AppWorx Handles System Failures

The AppWorx object-oriented architecture ensures orderly recovery after system failures.

If your system should fail for minutes, hours, or days, the AppWorx modular chains ensure orderly recovery. AppWorx keeps a record in its Oracle tables of each job and the job’s status. When you bring your system back up, AppWorx restores each job to the status at the time the system went down.

How AppWorx Handles Jobs in the Backlog

If the machine where the AppWorx master is installed goes down, AppWorx goes through a recovery procedure when the machine comes back up. The table below describes how AppWorx handles jobs that have already been submitted and are displayed in the Backlog.

<table>
<thead>
<tr>
<th>Job status at time of crash</th>
<th>Job status after recovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting or Running</td>
<td>AppWorx checks to see if the job process ID exists. If it exists, the status will be RUNNING. If the job process ID does not exist, the status will be DIED. You can restart the job from the Backlog.</td>
</tr>
<tr>
<td>Thread Wait</td>
<td>Status will be THREAD WAIT.</td>
</tr>
<tr>
<td>Date Pending</td>
<td>If the scheduled start date is still in the future, the status will be DATE PENDING. If the start date is past, AppWorx will launch the job as soon as possible after the system is brought back online.</td>
</tr>
<tr>
<td>Finished</td>
<td>The job will be moved to the History with a status of FINISHED.</td>
</tr>
<tr>
<td>Aborted</td>
<td>If the restart once on abort option is selected for the module, and this is the first time the job has aborted, AppWorx will run the job as soon as possible after the system is brought back online.</td>
</tr>
</tbody>
</table>
The Effect of Agent and Network Failures

The following table describes what happens when a job is running and the agent, agent machine, or network goes down.

<table>
<thead>
<tr>
<th>When a job is running and the agent</th>
<th>AND the machine running the agent</th>
<th>AND the network</th>
<th>THEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>is up</td>
<td>is up</td>
<td>is up</td>
<td>The status is RUNNING.</td>
</tr>
<tr>
<td>goes down</td>
<td></td>
<td></td>
<td>The status of the job in the Backlog is not changed, however the agent will continue to monitor the job. When the network comes back up, AppWorx checks with the agent and updates the status as RUNNING, FINISHED, or ABORTED.</td>
</tr>
<tr>
<td>goes down</td>
<td>is up</td>
<td></td>
<td>The status of the job in the Backlog is not changed. When the machine and agent come back up, AppWorx marks the job as DEAD. You can restart the job from AppWorx.</td>
</tr>
<tr>
<td>goes down</td>
<td>is up</td>
<td>is up</td>
<td>The status of the job in the Backlog is not changed. When the agent comes back up, AppWorx checks if the job completed. If the job completed, AppWorx reports the status. If the job did not complete, AppWorx will look for the job's PID. If it finds the PID, it will mark the status as RUNNING. If it does not find the PID, it will mark the status as DEAD.</td>
</tr>
</tbody>
</table>

How Schedules Impact Recovery

Modules and chains scheduled to run during the down time will run once when the system is brought back up. They will then return to their normal schedule. It does not matter how many times the module or chain was scheduled to run during the down time. Each module or chain will run only one time before returning to its normal schedule, unless a date was entered in the Reschedule from field. For information on schedules, see chapter 10: Scheduling Modules and Chains.

If the Oracle Tables are Lost

If the AppWorx Oracle tables are lost as a result of the system failure, all status information will be lost. You will need to restore the Oracle database and let AppWorx resume processing based on the schedule information restored by the backup.
6

Viewing Summaries and Forecasts

6.1 Introduction to Viewing Summaries and Forecasts ................................................ 120
6.2 Viewing Chain Summaries ..................................................................................... 122
6.3 Using the Gantt Chart in Chain Summaries ........................................................... 124
6.4 Working with a Chain Summary’s Flowchart ....................................................... 126
6.5 Printing Chain Summaries, Graphical Forecasts, and Chain Flows ....................... 128
6.6 Viewing a Forecast ................................................................................................. 130
6.7 Viewing a Graphical Forecast .............................................................................. 132
6.8 Setting the FORECAST Module Start Time(s) ..................................................... 134
6.9 Generating a Production Schedule ...................................................................... 136
6.1 Introduction to Viewing Summaries and Forecasts

With the Chain Summary and Forecast features, you can view chains and their components in the Backlog and History and view a list of scheduled modules and chains.

Viewing Chain Summaries

Using the Chain Summary feature you can select a chain or any of its components from the Backlog or History to view all the jobs belonging to it. You can also view the jobs belonging to all parent and child chains.

Note: The Chain Summary screen is not refreshed. To update its view you must close the Chain Summary window and reopen a new Chain Summary.

The Chain Summary window can be viewed in three ways depending on the option assigned to your license. For more information, see topic 1.6 AppWorx License Options. Your license key determines whether several ad-on features are included with your AppWorx master/agent instance.

If you have the Graphical Analysis Package option, the Graphical Forecast window includes an object tree, a table with a Gantt chart, and a flowchart (see Figure A).
For more information on the Chain Summary feature, see topic 6.2 Viewing Chain Summaries.

**Viewing Forecasts and Graphical Forecasts**

Using the Forecast feature, you can view a list of scheduled modules and chains (see Figure B). For more information on the Forecast feature, see topic 6.6 Viewing a Forecast.

If you have the Graphical Analysis Package option, you can view a graphical forecast of scheduled modules and chains. The Graphical Forecast window includes an object tree, a table with a Gantt chart, and a flowchart. It looks similar to the Chain Summary window shown in Figure A. For more information on the Graphical Forecast feature, see topic 6.7 Viewing a Graphical Forecast.

The data displayed in forecasts and graphical forecasts are generated and loaded into AppWorx by running the FORECAST module. When you create a schedule for the FORECAST module, you determine the timeframe of the forecast and how often it is run. For more information on running the FORECAST module, see topic 6.8 Setting the FORECAST Module Start Time(s).

**Viewing Production Schedules**

A detailed production schedule that shows conditions and parameters is available by running the PRODSCH chain. For information on running PRODSCH, see topic 6.9 Generating a Production Schedule.
6.2 Viewing Chain Summaries

To view a summary of modules running in a chain, right-click a chain or one of its components in the Backlog or History and select Chain Summary.

You can view a Chain Summary to see what components belong to a chain and its children.

**Procedure**

To view a summary of modules running in a chain:

1. Select a chain or one of its components in the Backlog or History and right-click.
   
   AppWorx displays the pop-up menu shown in Figure A.

2. Select the Chain Summary option.

   AppWorx displays the Chain Summary window shown in Figure B. The number on the title bar is the job ID of the chain or component you selected. The Chain Summary screen is not refreshed. To update its view you must close the Chain Summary window and reopen a new Chain Summary.

   **Note:** The Chain Summary may look different depending on the options assigned to your license. For more information, see topic 1.6 AppWorx License Options. The Chain Summary options are described later in the chapter.

   Each of the window’s panes can be expanded by dragging the splitter bars, or clicking their arrows. You can expand and collapse the chain icons in the tree and edit modules/chains by right-clicking. For more information on editing modules, see topic 5.8 Changing Job Statuses in the Backlog and topic 5.9 Viewing and Editing Job Details.

   You can maximize the Chain Summary window and the AppWorx desktop by going to the File menu and selecting Screen to full size.
**Chain Summary Column Descriptions**

The following information will be listed for each chain/component:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>Name of the queue.</td>
</tr>
<tr>
<td>Module</td>
<td>The alias name for the module or chain.</td>
</tr>
<tr>
<td>Job Started</td>
<td>The date and time the module is scheduled to start executing, after a job begins executing, the date and time it started.</td>
</tr>
<tr>
<td>Gantt chart</td>
<td>Used to show job execution statuses and times of chains and their components. It is included with the Graphical Analysis Package. For information on using the Gantt chart, see topic 6.3 Using the Gantt Chart in Chain Summaries.</td>
</tr>
<tr>
<td>Status</td>
<td>The current status of the module. For a list of status values, see Appendix B: Job Status Values.</td>
</tr>
<tr>
<td>Agent</td>
<td>The agent or agent group the module will execute on.</td>
</tr>
<tr>
<td>Requestor</td>
<td>The person that submitted the module.</td>
</tr>
<tr>
<td>Chain</td>
<td>The chain that contained the job.</td>
</tr>
<tr>
<td>Job ID</td>
<td>The identification number, which AppWorx auto-assigns to the job. <strong>Note:</strong> Decimal values at the end of Job IDs indicate that a job was restarted. A job has .01 added to its Job ID each time it is restarted.</td>
</tr>
<tr>
<td>Log</td>
<td>Uses a Y to identify jobs that includes operator logs.</td>
</tr>
</tbody>
</table>
6.3 Using the Gantt Chart in Chain Summaries

The Gantt chart is used to show job execution statuses and times of chains and their components.

If you have the Graphical Analysis Package option, the Chain Summary window includes a Gantt chart to show job execution statuses and run times of chains and their components (see Figure A). For more information on the Graphical Analysis Package, see topic 1.6 AppWorx License Options.

The Gantt chart shown on the right in the Chain Summary table in Figure A shows the following:

- Blue boxes show the estimated execution time of each chain/component.
- Bars show the actual execution time of each chain/component. The bar is color coded using the same scheme as the Status Bar.

  For information on the Status Bar, see topic 5.4 Monitoring with the Status Bar and Object Icons.

- A dotted vertical line shows the current time.

If your chain has run for a long period of time you can scroll through the Gantt chart using the Gantt Scroll bar, and/or select a time from the Visible Hours drop-down box.
6.4 Working with a Chain Summary’s Flowchart

You can expand and collapse the chains in the flowchart and select a job to bring it into focus in the table and the tree. You also resize the display and choose whether to have AppWorx display predecessor relationships, execution order, and conditions.

When you select a chain or component in the flowchart, AppWorx displays a blue border around it. AppWorx also brings it into focus in the summary and the tree. In Figure A the LEVEL_5 chain is selected. You can edit modules/chains in the summary or tree by right-clicking. For more information on editing modules, see topic 5.8 Changing Job Statuses in the Backlog and topic 5.9 Viewing and Editing Job Details.

Expanding and Collapsing Chains

You can expand or collapse a chain in the flowchart by clicking the ‘+’ or ‘-’ buttons in their title bar(s).
Resizing the Display

AppWorx automatically resizes the display when you expand or collapse a chain. You can also resize the flowchart’s display in the following ways:

- Expand the viewable area with the splitter bars, by dragging them or clicking their arrows.
- Use the **0-200** percentage scale to manually adjust the scale.
- From the flowchart’s View menu, select from the following boxes to automatically adjust the display:
  - Fit to size
  - One time fit to size
  - Show vertical (orients flowchart vertically)
  - Show horizontal (orients flowchart horizontally)

Display Options

From the flow chart’s View menu, you can use the check boxes to choose whether to have AppWorx display:

- Predecessor relationships and execution order (using the Show Predecessors, Show Order, Show all arrows, and Show no arrows options).
- Only the relationships to the module/chain selected with the mouse (using the Draw to/from dependencies only option).
6.5 Printing Chain Summaries, Graphical Forecasts, and Chain Flows

You can print, preview, or set up a print job using the File menu. You select print options on the Page Setup window.

From the Chain Summary, Graphical Forecast, and Chain Flow windows you can go to the File menu and select an option to:

- Print the tree, table, or flowchart. When you print the flowcharts, you can decide whether to print the portion of the flowchart which is in view or the entire flowchart on several pages.
- Set up options for a print job.
- Preview a print job.

A Chain Summary window is shown in Figure A.

Setting Up Print Options

Before printing, you can go to the File menu and select Page Setup to select print options. From the Page Setup window shown in Figure B, you can select various print options.

Note: Any options you select by clicking the Properties button on the Print window (see Figure C) are not applied. Only the options selected in the Page Setup window are used.
Figure B. The Page Setup window.

Figure C. Select print options on the Page Setup window only.
6.6 Viewing a Forecast

To view a list of scheduled modules and chains, go to the View menu and select Forecast.

To view a list of scheduled modules and chains, go to the View menu and select Forecast (see Figure A).

Data displayed in forecasts is generated and loaded into AppWorx by running the FORECAST module. When you create a schedule for the FORECAST module, you determine the timeframe of the forecast and how often it is run. For more information on running the FORECAST module, see topic 6.8 Setting the FORECAST Module Start Time(s).

To view a forecast, go to the View menu and select Forecast (see Figure A). AppWorx displays the printable list of modules and chains that are scheduled to run through the end of the next day (see Figure B). Each scheduled module/chain includes the start date and time and the module or chain’s name. Chains also include a key icon used to expand/collapse them. Each chain component includes:

- The component’s order in the chain.
- An (S) or (M) for single or multi-threaded.
- The component’s name or alias.

To view the modules within a chain, click the chain’s key. To expand all keys for a chain and its children, select the chain, go to the View menu and select Expand Chains. To expand the keys in all chains, choose Expand All.
Viewing Graphical Forecasts

If you have the Graphical Analysis Package option, you can view a graphical forecast of scheduled modules and chains. For information, see topic 6.7 Viewing a Graphical Forecast.

Viewing a Production Schedule

A more detailed production schedule that shows conditions and parameters is available by running the PRODSCH chain. For information on running PRODSCH, see topic 6.9 Generating a Production Schedule.
6.7 Viewing a Graphical Forecast

If you have the Graphical Analysis Package option, you can view a graphical forecast of scheduled jobs.

If you have the Graphical Analysis Package option you can view a graphical forecast to see what jobs are scheduled to run. For more information on the Graphical Analysis Package, see topic 1.6 AppWorx License Options.

The data displayed in a graphical forecast is generated and loaded into AppWorx by running the FORECAST module. When you create a schedule for the FORECAST module, you determine the timeframe of the forecast and how often it is run. For more information on running the FORECAST module, see topic 6.8 Setting the FORECAST Module Start Time(s).

To view a Graphical Forecast, go to the View menu and select Graphical Forecast. AppWorx displays the Chain Flow Forecast window shown in Figure A. It has three panes which include:

- A tree of all scheduled modules and chains.
- A table with the details of each module/chain. The table includes a Gantt chart used to show estimated run times of chains and their components.
- A graphical display of all scheduled modules and chains.
If you select a chain from the tree or graphic display, AppWorx displays its components in the table.

You can maximize the Graphical Forecast window and the AppWorx desktop by going to the File menu and selecting Screen to full size.

**Expanding and Collapsing Chains**

You can expand or collapse a chain in the tree by clicking the chain’s key. You can expand or collapse a chain in the flowchart by clicking the ‘+’ or ‘-’ buttons in their title bar(s).

**Resizing the Display**

AppWorx automatically resizes the display when you expand or collapse a chain. You can also resize the flowchart’s display in the following ways:

- Expand the viewable area with the splitter bars, by dragging them or clicking their arrows.
- Use the 0-200 percentage scale to manually adjust the scale.
- From the flowchart’s View menu, select from the following boxes to automatically adjust the display:
  - Fit to size
  - One time fit to size
  - Show vertical (orients flowchart vertically)
  - Show horizontal (orients flowchart horizontally)

**Display Options**

From the flow chart’s View menu, you can use the check boxes to choose whether to have AppWorx display:

- Predecessor relationships and execution order (using the Show Predecessors, Show Order, Show all arrows, and Show no arrows options).
  Only the relationships to the module/chain selected with the mouse (using the Draw to/from dependencies only option).
6.8 Setting the FORECAST Module Start Time(s)

The data displayed in a forecast or graphical forecast is generated and loaded into AppWorx by running the FORECAST module.

The data displayed in a forecast or graphical forecast (see Figure A) is generated and loaded into AppWorx by running the FORECAST module. When you create a schedule for the FORECAST module, you determine the timeframe of the forecast and how often it is run.

![Figure A. The graphical forecast](image)

Depending on your organization's needs, you can schedule this module to run every day, or several times each day.

Additionally, if you want to set the forecast on an ad hoc basis, you can submit the module as shown in Figure B.

The SET_VIRTUAL_DAY module includes one editable prompt. The default value for the prompt is a series of numbers that represent the current date and time.
In Figure C the default value is 20020219090643, this translates to:

- Year: 2002
- Month: February (02)
- Day of month: 19th
- Time: 9:06:43 A.M.

Figure B. You can submit the FORECAST module to reset the forecast data.
6.9 Generating a Production Schedule

To generate a production schedule report, run the PRODSCH chain and view output for the SCHPRINT job.

If you want to see a schedule of all modules and chains that are scheduled to run between specified dates, you can run the PRODSCH chain. The chain runs two modules SCHCREATE and SCHPRINT. The output for the SCHPRINT module reports the jobs by date and time. It includes the following information:

- The date and time each chain is scheduled to run.
- The name of each module in each chain.
- The conditions associated with each module.

The report can show only the jobs that will run, or all jobs that will run and all jobs that are eligible to run but will not run due to conditions set on the module. The sample production schedule report shown in this topic displays two chains: SYSTEM and SALES_REPORTS. The Skip column in the analyst’s report displays an abbreviation indicating why the module will not be run. Notice that NDOW displays for HISTORY_PURGE. Chain names are in [brackets]. Module names follow the chain names. The SALES_REPORTS chain runs three modules: REGION_A, REGION_B, and REGION_C. The first two modules each have a BEFORE condition.

Generating a Production Schedule

To generate a production schedule report:

1. Request and submit the PRODSCH chain.
   The chain runs two modules SCHCREATE and SCHPRINT.
2. Enter the start and end dates for the report.
3. Choose Yes or No for the analyst’s status values.
   Choose Y to show all jobs including those that will not run due to the days of the week settings and conditions. This is useful for analysts that are reviewing schedules to make sure they will do what was intended. Choose N to display only those jobs that will run. This report is most useful for operators who are monitoring the system.
4. Enter the minimum schedule number.

<table>
<thead>
<tr>
<th>To display</th>
<th>Use this value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jobs scheduled to run daily</td>
<td>-3</td>
</tr>
<tr>
<td>Jobs scheduled to run hourly</td>
<td>-4</td>
</tr>
<tr>
<td>Jobs scheduled to run on minute intervals</td>
<td>-5</td>
</tr>
</tbody>
</table>

**Note:** If selecting an interval other than -3, you should review your start and end dates, because the shorter interval settings will produce larger reports.

5. Submit the module and view the output for the SCHPRINT module.

**Sample Production Schedule**

```plaintext
Skip {Chain}Report Name
--- ----------------------------------------------------
Saturday Feb 23 2002 00:00
  {SYSTEM}Saturday Feb 23 2002 00:00
  {SYSTEM}DELDEFAULT
NDOW {SYSTEM}HISTORY_PURGE

Monday Feb 25 2002 00:00
  {SALES_REPORTS}Monday FEB 25 2002 00:30
  {SALES_REPORTS}REGION_A
    B If CURRENT TIME > 06:00:00 then SKIP TASK
  {SALES_REPORTS}REGION_B
    B If CHECK FILE NO /reports/region_b.dat
  {SALES_REPORTS}REGION_C
```

**Production Schedule Output Abbreviations**

The five abbreviations used in the production schedule are described below.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACT</td>
<td>Not active</td>
</tr>
<tr>
<td>NDOW</td>
<td>Not run, day of week</td>
</tr>
<tr>
<td>SONNHD</td>
<td>Skip, not in calendar (being run with a calendar and it’s not in the calendar)</td>
</tr>
<tr>
<td>RONHD</td>
<td>Run, in calendar (being run with a calendar and it’s in the calendar)</td>
</tr>
<tr>
<td>SONHD</td>
<td>Skip, in calendar (skip using a skip calendar)</td>
</tr>
</tbody>
</table>
Using the AppWorx Command Line Interface

7.1 Introduction to the AppWorx Command Line Interface .................................................. 140
7.2 Entering Generic Commands ....................................................................................... 142
7.3 Entering Backlog Commands ..................................................................................... 146
7.4 Entering Palm Commands ......................................................................................... 150
7.5 Entering Other Commands ....................................................................................... 152
7.1 Introduction to the AppWorx Command Line Interface

With the AppWorx command line interface you can manage masters, agents, queues, processes, and jobs.

You can access all the functionality of Jobs/Explorer window with the AppWorx command line interface.

There are two ways to enter AppWorx command line functions in command mode, or as a single function. An example is shown for each executing the 'h' help command.

- Type **appworx** at the command line, to go into command mode.
  
  aix433:/home/appworx50>appworx
  AppWorx >h

- Type **appworx command**, to execute a single function.
  
  aix433:/home/appworx50>appworx h

The help command displays a list of command line functions. The command line functions are shown in the table below, the subheadings that follow describe each command and give examples.

**Generic Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle agentname [Agent]</td>
<td>Idle an agent or master</td>
</tr>
<tr>
<td>RESume agentname [Agent]</td>
<td>Resume an agent or master</td>
</tr>
<tr>
<td>RUN module_name [options] [args]</td>
<td>Request a module to run</td>
</tr>
<tr>
<td>STARtso agent(ALL) [Agent]</td>
<td>Start AppWorx processes</td>
</tr>
<tr>
<td>STATus</td>
<td>Process status list</td>
</tr>
<tr>
<td>STOPso agent(ALL) [Agent]</td>
<td>Stop AppWorx processes</td>
</tr>
<tr>
<td>User</td>
<td>Process Listing of user soport</td>
</tr>
<tr>
<td>Watchworx start</td>
<td>stop</td>
</tr>
</tbody>
</table>

**Backlog Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete jobid(s)</td>
<td>Delete selected jobids</td>
</tr>
<tr>
<td>HOLD jobid(s)</td>
<td>Hold selected jobids</td>
</tr>
<tr>
<td>Kill jobid(s)</td>
<td>Kill selected jobids</td>
</tr>
<tr>
<td>Release jobid(s)</td>
<td>Release from Hold selected jobids</td>
</tr>
<tr>
<td>Getvar varname</td>
<td>Retrieve Static Subvar</td>
</tr>
<tr>
<td>Setvar varname=value</td>
<td>Set Subvar static value</td>
</tr>
<tr>
<td>JQ</td>
<td>Job Queue Listing</td>
</tr>
<tr>
<td>JH [HISTORY=mins]</td>
<td>Job History Listing</td>
</tr>
<tr>
<td>Agents</td>
<td>Agent Summary</td>
</tr>
<tr>
<td>Queue</td>
<td>Queue Summary</td>
</tr>
</tbody>
</table>
Palm Commands

PQ  Shortened Job Queue Listing
PH [minutes]  Shortened Job History Listing
PAgents  Shortened Agents Summary

Other Commands
Load filename  Loads file and extracts 1st field as jobids Can be used by Delete, Kill, Hold, Release. (I.E del jobids)
! {number}  Re-execute command (I.E. !12)
! {command}  Host Command (I.E. !who)
| {command}  Pipe Host Commands (I.E. jq|grep BATCH)

HISTORY  Display command history
Help  Display this help screen
eXit  Exit AppWorx
QUIt  Quit AppWorx

Notice that each of the command line functions are listed beginning with one to four capital letters. These are the only required characters for the command. So, it is okay to enter Q (or Qu, Que, or Queu if you like) for Queue. Functions are not case sensitive, so q, qu, que, queu, and queue will work just as well.

Note: While in command mode, you can press Enter to execute the previous command again.

Assigning Users to an Exe Range

In order for users to use the AppWorx CLI, they must have them assigned in the Awexe range. If you enter 2000-2099, the user will have access to all the AppWorx Command Line Interface (CLI) functions. However to change any data, they will be required to supply a AppWorx user name and password. For more information on setting the Awexe range, see topic 16.2 Defining Users.

Restricting Users

The -restricted command restricts the CLI from executing shell commands and pipe commands. This is for creating restricted login accounts that will run the CLI from a captive login script.
7.2 Entering Generic Commands

Each of the generic AppWorx command line functions are described with examples in this topic.

The generic AppWorx command line functions are used to:
• Idle and resume masters and agents.
• Run modules.
• Start and stop AppWorx processes.
• List major process that are running.
• List all processes that are running for a user.
• Start or stop the local WatchWorx process.

Idling and Resuming Agents and Masters

To idle an agent or master, type the following:

```
Idle agentname [Agent]
```

To resume an agent or master, type the following:

```
RESume agentname [Agent]
```

Example: Idling an Agent

To idle the agent PROD-HP:

```
AppWorx >i PROD-HP
```

Example: Idling a Master

To idle the master PROD-HP:

```
AppWorx >idle PROD-HP M
```

Example: Resuming an Agent

To resume the agent PROD-HP:

```
AppWorx >res prod-hp
```

Example: Resuming a Master

To resume the master PROD-HP:

```
AppWorx >resume prod-hp
```
Running Modules

To run a module, type the following:

```
RUN module_name [options] [args]
```

You can type ‘run’ and press Enter, to view a list of options. The run function reflects awrun. For more information, see Appendix F: awrun – The AppWorx Command Line Utility in the Technical References Guide.

Starting and Stopping Commands

To start an agent and/or master, type the following:

```
STARTso agent(ALL) [Agent] Start AppWorx processes (Default=ALL)
```

To stop an agent and/or master, type the following:

```
STOPso agent(ALL) [Agent] Stop AppWorx processes (Default=ALL)
```

**Example: Starting a Master and Agent**

To start the master and the agent PROD-HP:

```
AppWorx >start PROD-HP
```

**Example: Stopping a Master and Agent**

To stop the master and the agent PROD-HP:

```
AppWorx >stop prod-hp
```

**Example: Stopping an Agent**

To stop just the agent PROD-HP:

```
AppWorx >stop prod-hp agent
```

or:

```
AppWorx >stop prod-hp a
```

**Example: Starting a Master**

To start just the master PROD-HP:

```
AppWorx >start prod-hp master
```

**Example: Starting a Master and All its Remote Agent**

To start the master and all local and remote agents:

```
AppWorx >start
```

You will be prompted with the following question where you must type ‘y’ for yes.

```
'startoper ALL ' Are you sure? y
```
Example: Stopping a Master and All its Remote Agent

To stop the master and all local and remote agents:

```
AppWorx >stop
```

You will be prompted with the following question where you must type 'y' for yes.

```
'stopoper ALL ' Are you sure? Y
```

Viewing Process Statuses

To view a list of process statuses, type the following:

```
STATus
```

Example

```
AppWorx >stat
```

Listing of Major Processes

- SUN26 awcomm Online for 4 hours (0)
- PROD-HP awcomm Online for 5 hours (0)
- PROD-HP awtcpd Online for 22 hours (0)
- PROD-HP awapi Online for 13 seconds (60)
- SUN27DEV awapi Online for 7 seconds (60)
- PROD-HP rmi Online for 2 days (0)
- PROD-HP watchworx Online for 22 hours (0)
- PROD-HP awmaster Offline for 164 seconds since 8/20/2001 13:10:41
- APPS awoae Online for 15 seconds (60)
- PROD-HP awagent Offline for 166 seconds since 08/20/2001 13:10:40
- SUN27DEV awagent Online for 9 seconds (60)

Process Listing of User soport

To view a process listing of user soport, type the following:

```
User
```

Example

```
AppWorx>user
soport 10386 10384 0 Aug 17 ? 0:00 /isa/apache/bin/httpd -d /isa/devel/soport/dev/v
soport 22401 1 0 Aug 18 ? 34:24 /isa/j2rel_3_0/bin/../bin/sparc/native_threads/java -
soport 15194 1 0 Aug 12 ? 0:04 /isa/devel/soport/dev/c/awcomm APPWORX
```
Starting and Stopping the WatchWorx Process

To start the local WatchWorx process, type the following:

`Watchworx start`

To stop the local WatchWorx process, type the following:

`Watchworx stop`
7.3 Entering Backlog Commands

Each of the Backlog AppWorx command line functions are described with examples in this topic.

The Backlog AppWorx command line functions are used to:

- Change job statuses in the Backlog.
- Get and set substitution variables.
- View a summary of the Backlog, History, agents, or queues.

Deleting Jobs in the Backlog

To delete selected jobids in the Backlog, type the following:

Delete jobid(s)

Example

AppWorx >del 680942

You will be prompted with to enter a User Id and password as shown below:

You have requested a command which requires a user_id and password before execution

Userid (soport)>sqloper
Password ()>

When you delete the jobid, you will see the following message:

Success 1 jobids processed

Killing Jobs in the Backlog

To kill selected jobids in the Backlog, type the following:

Kill jobid(s)

Putting Jobs On Hold in the Backlog

To hold selected jobids in the Backlog, type the following:

Hold jobid(s)

Example

AppWorx >hold 680941 680940

When you put the jobid on hold, you will see the following message:

Success 2 jobids processed
Releasing (Resetting) Jobs in the Backlog
To release from Hold selected jobids in the Backlog, type the following:

Release jobid(s)

Retrieving Static Substitution Variables
To view the value of a static substitution variable, type `Getvar varname`.

Example
This example returns the value of the subvar `#company_name`.

```
AppWorx >G #company_name
ABC Corporation
```

Setting Static Substitution Variables
To set the value of a static substitution variable, type `Setvar varname=value`.

Example
This example sets the value of the `#company_name` substitution variable to ABC Inc.

```
AppWorx >S #company_name=ABC Inc.
AppWorx >G #company_name
ABC Inc.
```

Viewing Jobs in the Backlog
To view a list of jobs in the Backlog, type the following:

`JQ`

Example
```
AppWorx >jq|grep M21
M21 680941.00 BATCH M21 08/19 10:07 NULL APPWORX PROGRAMMERLOADCHAIN
```

Viewing Jobs in the History
To view a list of jobs in the History, type the following:

`JH [HISTORY=mins]`
Viewing an Agent Summary

To view an agent summary, type the following:

Agents

Example

```
AppWorx >a
Agent Status Last Act M/A Thds Bk Run Hld Abt Sch
PROD-HP Stopped 00:40:04 M 99 34 0 11 0 27
APPS Running 00:00:50 A 99 0 0 0 0 0
PROD-HP Stopped 00:40:39 A 99 34 0 11 0 27
SUN27DEV Running 00:00:23 A 99 0 0 0 0 0
ALL G 0 0 0 0 0 0
OAEGROUP G 0 0 0 0 0 0
```

Viewing a Queue Summary

To view a queue summary, type the following:

Queue

Example

```
AppWorx >q
Queue Thds Bk Stat Pri Sched
BATCH 999 72 A 50 59
BURT_AND_E 1 0 A 49 1
EXPRESS 0 0 I 10 99
GIANG 1 0 A 50 1
MONITOR 99 0 A 50 99
NEWQUEUE 999 0 A 59 59
PRIORITY 1 0 A 40 1
Total Qs 99 72 A 99
```
7.4 Entering Palm Commands

Each of the palm AppWorx command line functions are described with examples in this topic.

As part of the AppWorx Anywhere solution, palm versions of three Backlog Commands allow you to use a PDA to view a summary of:

- Jobs in the Backlog
- Jobs in the History
- Agents.

Viewing Jobs in the Backlog with a PDA

To view a list of jobs in the Backlog using a PDA, type the following:

```
PQ
```

**Example**

```
AppWorx >PQ
Module                      Status    Jobid
PREDECESSOR_DEMO INITIATED 148171
QA_DAYS                     INITIATED 154167
TIMING_CHAIN INITIATED 154007
TIMING_CHAIN INITIATED 154214
CHAIN_REPORT NULL 153928
CHAIN_REPORT PRED WAIT 153945
```

Viewing Jobs in the History with a PDA

To view a list of jobs in the History using a PDA, type the following:

```
PH
```

Viewing an Agent Summary with a PDA

To view an agent summary using a PDA, type the following:

```
PA
```

**Example**

```
AppWorx >pa
Agent   Status  M/A Bk Run Hld Abt Sch
SUN26X Running M 36 0 10 5 12
SUN27DEV Running A 0 0 0 2 0
SUN26X Running A 32 0 9 1 12
OAE Stopped A 0 0 0 0 0
ALL       G 2 0 0 1 0
```
7.5 Entering Other Commands

Each of the other AppWorx command line functions are described with examples in this topic.

The other AppWorx command line functions are used to:
- Load filenames (to execute commands).
- Re-execute commands.
- Execute host commands.
- Pipe host commands.
- Display a command history.
- View the AppWorx Command Line help.
- Exit the AppWorx command mode.

Loading a File

To load a file from AppWorx command mode, type **Load filename.**

**Example**

In this example, we grep the job queue (Backlog) and find six modules whose name contains “SKIP”. We redirect the output to the file **xx**. Next, we load **xx**, and enter the **del jobids** command to delete the jobids. Since this command requires a user id and password, we must enter them. AppWorx displays the jobid numbers, and we are asked whether we are sure we want to delete these files.

```
AppWorx> jq|grep SKIP
Jobid Queue Module Start Date Status Agent Requestor Chain
683549.00 BATCH SKIPS_MOND09/13 00:30 Monday APPWORX QA_DAYS
683550.00 BATCH SKIPS_TUES09/13 00:30 Tuesday APPWORX QA_DAYS
683551.00 BATCH SKIPS_WEDN09/13 00:30 Wednesday APPWORX QA_DAYS
683552.00 BATCH SKIPS_THUR09/13 00:30 Thursday APPWORX QA_DAYS
683553.00 BATCH SKIPS_SATU09/13 00:30 Saturday APPWORX QA_DAYS
683554.00 BATCH SKIPS_SUND09/13 00:30 Sunday APPWORX QA_DAYS
AppWorx> jq|grep SKIP>xx
AppWorx> load xx
jobids= 683549.00 683550.00 683551.00 683552.00 683553.00 683554.00
AppWorx> del jobids
You have requested a command that requires a user id and password before execution
User id [soport]
Password [ ]
Delete 683549.00 683550.00 683551.00 683552.00 683553.00 683554.00
Are you sure? n
```
Running a Host Command

To run a host command from AppWorx command mode, simply type `!{command}`.

Example

To identify users currently logged in.

```
AppWorx > !who
```

Displaying a Command History and Re-Executing Commands

To display a history of commands, type `hi`.

Example

```
AppWorx > hi
1  jh
2  jq
3  jq|grep INIT
4  del 682255
5  hi
```

To re-execute a command, type `!{number}`.

Example

```
AppWorx> !3 re-executing: jq|grep INIT
Jobid Queue Module Start Date Status Agent Requestor Chain
682253.00 BATCH QA_DAYS 08/31 00:31 INITIATED APPWORX
682254.00 BATCH QA_CHAINS 08/31 00:31 INITIATED APPWORX SQLOPER
682281.00 BATCH QA_DAYS 08/31 00:31 INITIATED APPWORX SQLOPER
   QA_CHAINS
682308.00 BATCH SCHED-TIME 08/31 05:00 INITIATED APPWORX
682309.00 BATCH QA_DAYS 08/31 05:00 INITIATED APPWORX TIME_ZONE_TEST
```

Piping Host Commands

To connect a host command to an AppWorx command function, type a pipe symbol `|`.

Example

To search for modules in the Backlog, which are running on the BATCH queue.

```
AppWorx > jq|grep BATCH
```

Exiting the AppWorx Command Line Interface

To exit AppWorx Command Line Interface, type `eXit` or `QUIT`.
Section III

Development

Chapter 8: Creating Modules .......................................................................................... 157
Chapter 9: Creating Chains ............................................................................................ 189
Chapter 10: Scheduling Modules and Chains ................................................................. 225
Chapter 11: Adding Prompts to Modules and Chains ...................................................... 247
Chapter 12: Assigning Predecessor Statements ............................................................... 281
Chapter 13: Working with Conditions ............................................................................ 299
Chapter 14: Defining Substitution Variables .................................................................. 345
8

Creating Modules

8.1 Introduction to Creating Modules ................................................................. 158
8.2 Defining Modules .......................................................................................... 160
8.3 Updating Unsaved Changes ......................................................................... 162
8.4 Entering General Information for Modules .................................................. 164
  8.4.1 Entering Program Information ................................................................. 166
  8.4.2 Defining Applications ............................................................................. 168
  8.4.3 Defining Libraries .................................................................................. 170
  8.4.4 Defining Program Types ......................................................................... 172
  8.4.5 Commonly Used Program Types That Ship with AppWorx ..................... 174
  8.4.6 Entering Execution Options .................................................................... 176
8.5 Specifying Output Options for Modules ...................................................... 178
8.6 Setting Module Logins .................................................................................. 180
8.7 Adding Notes ................................................................................................ 182
8.8 Assigning Roles to Modules ......................................................................... 184
8.9 Copying Modules .......................................................................................... 186
8.1 Introduction to Creating Modules

A module contains all the information required to execute a program and handle its output. Modules can be run with a schedule or on an ad hoc basis. They can also be included in an AppWorx chain. You can add a module to as many chains as you wish. If you change a module definition, the change is applied to every chain that includes the module.

A module is the basic building block in AppWorx. For each program you want to run (for example: FTP, application, database load), you must create a module. A module specifies all the information required to run a program including:

- General information: The task that will be run, its program information and execution options.
- Output and login source options.
- Prompts: Information that is passed to the program as variables.

Modules can be run individually with a schedule or on an ad hoc basis by from the Requests window. They also can be run as a component of an AppWorx chain. The Modules window is shown in Figure A, with the CHAIN_REPORT module displayed.

![Figure A. The General tab for modules.](image-url)
How Modules Fit into AppWorx

If you have been using scripts to run your operations, you have launched programs from within a script. In AppWorx, you create a module to launch each program. After creating a module, you can run the module by itself, or add it to a chain. Figure B shows the relationship between scripts, modules, and chains.

Keeping with the AppWorx object-oriented approach to operations, you can use a module in as many different chains as you wish. If you change a module definition, the change is applied to every chain that includes the module. You do not have to change the definition in each chain. This saves you a great deal of time maintaining your system.
8.2 Defining Modules

To add a module to AppWorx, go to the Object Admin menu on the desktop and select Modules. Click New and select a module type.

To add a module to AppWorx, go to the Object Admin menu on the desktop and select Modules.

![Figure A. The Modules window.](image)

**Note:** AppWorx roles control access to modules. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To add a module to AppWorx:

1. From the desktop, open the Modules selector window in one of two ways:
   - Select the modules icon from the toolbar.
   - Go to the Object Admin menu and select Modules.
   
   For more information, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. From the Modules selector window, click **New**.
   
   AppWorx displays the Select module type window shown in Figure B.
3. Select a module type. The module type you select will determine the default selections for certain fields, such as the program type and login.

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Used to run scripts and execute programs. This is the module type you will select most of the time.</td>
</tr>
<tr>
<td>OAE</td>
<td>Used by the AppWorx Oracle Applications Extension to interface with Oracle E-Business Suite' Concurrent Manager.</td>
</tr>
<tr>
<td>SAP</td>
<td>Used by the AppWorx mySAP.com Extension to interface with the mySAP.com Extension.</td>
</tr>
<tr>
<td>Schedule</td>
<td>A special module that is used in unusual circumstances to call a chain on an ad hoc basis. <strong>Note:</strong> In prior versions of AppWorx, schedule modules were needed to submit any chain on an ad hoc basis. With version 5.1, chains are included in the Requests window. For more information, see topic 9.9 Why Use a Schedule Module?</td>
</tr>
</tbody>
</table>

AppWorx opens the Module window shown in Figure A.

4. Define the module by assigning values to the fields on the General tab. For more information, see topic 8.4 Entering General Information for Modules.

**Editing and Deleting Modules**

To edit or delete a module, select the module on the Modules selector window and select the appropriate button. For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

If a module is used in one or more chains, you must remove it before you can delete the module. If a module is used in a chain and you try to delete it, AppWorx will display a message saying the module is in use.

If a module is in the Backlog, you cannot delete it. You will need to wait until the job has completed executing before you can delete its definition.
8.3 Updating Unsaved Changes

When you enter or change a field value, AppWorx displays a triangle next to the current tab’s label to signify unsaved changes. A red triangle signifies changes made to a sub-element of a tab. You must update these sub-elements to save their changes.

Sub-elements of a tab include:

- Sub-tabs on the Modules and Chains windows.
- A table with Add, Edit, and Delete buttons. These tables are used when defining thread schedules and printer distribution lists.

Figure A. Triangles display unsaved changes to tabs and their sub-elements.

In the Modules window shown in Figure A, unsaved changes have been made on the Schedule and Predecessors tabs as well as the Schedule’s General and Frequency sub-tabs.

The Schedule tab includes a red triangle. You cannot update the module’s definition in the database until you update the Schedule tab.
If update click **OK** or **Save**, AppWorx displays the window in Figure B. From this window you can click **Yes** to discard the pending changes for the tabs listed, or **No** to remain on the tab in the object’s definition.

In order to update the Schedule tab without bringing up the window in Figure B, you must add the schedule, by clicking **Add**, or clear it by clicking **Clear**.

---

**Figure B.** Click **No** to remain on the tab.
8.4 Entering General Information for Modules

To define a module you must give it a name and description. And assign it to; an application, an agent, program information, and execution options.

For a module to execute a program, you must define the program on the General tab of the Modules window shown in Figure A. This information tells AppWorx where to find the program the module will run and how to run it.

![Figure A.](image)

You can define certain objects by clicking the icon to the right of their field. AppWorx will display an edit window where you can define the new object. Fields marked with an “*” on their right are required.

**Procedure**

To define the program that a module will run:

1. Enter a name and description for the module on the General tab of the Modules window shown in Figure A.
   The entries may be up to 30 characters long each.
   **Note:** Do not name modules ‘SCHED-<chain name>‘, unless they are schedule modules. Standard modules with these names will not run.

2. Select an application.
   Applications are the general category to which modules and chains belong (for example: inventory, accounts payable). For more information on defining applications, see topic 8.4.2 Defining Applications.
3. Select the agent or agent group, where the program you wish to run is stored and where it will run from.

4. Assign values to the fields in the Program and Execution options boxes. These fields are described in topic 8.4.1 Entering Program Information and topic 8.4.6 Entering Execution Options.

5. If you wish, you can select the other tabs on the Modules window to edit them at this time.

6. Click Apply to add the module definition to the list of modules, or OK to add the new definition to the list of modules and close the Modules window.

   AppWorx displays an error message to alert you to any unspecified fields when they need to be filled in.

**Subtopics**

For information on entering program information and execution options, and defining program objects, see the following subtopics:

- 8.4.1 Entering Program Information
- 8.4.2 Defining Applications
- 8.4.3 Defining Libraries
- 8.4.4 Defining Program Types
- 8.4.5 Commonly Used Program Types That Ship with AppWorx
- 8.4.6 Entering Execution Options
8.4 Entering General Information for Modules

8.4.1 Entering Program Information

In order for a module to execute a program or run a chain, you must define its program information (a library, program type, and program name). You enter program information on the General tab of the Modules window. When the program information has been assigned, AppWorx displays the path and file name in the non-editable Location field.

In order for a module to execute a program or run a chain, you must define its program information in the Program box (shown in Figure A) on the General tab.

![Figure A. The Program box, found on the General tab of the Modules window.](image)

The Program fields are described in the table below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>The path to the program source or executable. The program type defined in the Type field may supply a subpath. For information on defining libraries, see topic 8.4.3 Defining Libraries.</td>
</tr>
<tr>
<td>Type</td>
<td>This is the host shell, or other command type that is defined for your system. AppWorx comes with several program types already defined. For more information on defining program types, see topic 8.4.4 Defining Program Types.</td>
</tr>
<tr>
<td>Program name</td>
<td>The name of the program that will be executed by the module. Note: If the program type selected is SCHEDULE, AppWorx displays a list of chains instead of programs. For more information, see topic 9.9 Why Use a Schedule Module?</td>
</tr>
</tbody>
</table>
When the program information has been assigned, AppWorx displays the path and file name in the non-editable **Location** field.
8.4 Entering General Information for Modules

8.4.2 Defining Applications

You use applications to group modules and chains. When defining or requesting modules and chains, you can select an application. To define an application, you must give it a name and description.

The **Application** field provides a way to assign modules to an organizational group. The groups might be programs, machines, divisions, or some other characteristic. When editing or requesting modules and chains, you can select an application, the modules and chains displayed are limited to those that are assigned to the application.

**Note:** AppWorx roles control access to applications. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To define a new application:

1. From the Applications selector window, click **New**.
   AppWorx opens the Applications window shown in Figure A.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.
2. Enter an application name and description.
8.4 Entering General Information for Modules

8.4.3 Defining Libraries

A library defines the path for the program a module runs. To define a library, you must name the library and enter its path.

When you create a module, you specify the program to be run by the module. In order for AppWorx to find the program, you must specify the library for the program. The library defines the path for the program.

You can update library definitions at any time. The updates are applied immediately to all modules that use the definition.

**Note:** AppWorx roles control access to libraries. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To define a new library:

1. From the Libraries selector window, click **New**.
   
   AppWorx opens the Libraries window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Enter a library name and path.
   
   The name is a short identifier for the path.
   
   The path can be an environment variable or an actual path. If you use an environment variable other than $AW_HOME, you must define the variable for each instance of AppWorx. For UNIX machines, define the variable at the end of the $AW_HOME/site/sosite file in Bourne Shell format. To define a variable called INVENTORY_APP, you would enter:

   INVENTORY_APP=/inventory/programs; export INVENTORY_APP
For Windows instances, define the variable at the end of `site\envvar.bat` as follows:

```
set INVENTORY_APP=\inventory\program
```
8.4 Entering General Information for Modules

8.4.4 Defining Program Types

A program type defines how a program accepts input and handles output. To define a program type, you must enter a name, description, host command, command path, parameter format (if appropriate), and file extension.

A program type defines how a program accepts input and handles output. AppWorx comes with several predefined program types (see topic 8.4.5 Commonly Used Program Types That Ship with AppWorx). You can define as many of your own program types as needed. Each program type is supported by a program type script located in $AW_HOME/exec. The program type script defines the method for passing input to the program and getting output from the program.

![Figure A. The Program Types window.](image)

Note: AppWorx roles control access to program types. If you do not have access to them, see your AppWorx administrator.

Procedure

To define a new program type:

1. From the Program Types selector window, click New.
   AppWorx opens the Program Types window shown in Figure A.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.
2. Fill in the fields described in Table A.
Table A. The Program Types window fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Name of the program type. Field length: 10 characters.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the program type. Field length: 30 characters.</td>
</tr>
<tr>
<td>Host command</td>
<td>Name of the program type script located in $AW_HOME/exec that supports the program type. The program type script defines the method for passing input to the program and getting output from the program.</td>
</tr>
</tbody>
</table>
| Param format  | Describes the format of data passed to the program. Usually #v=#d, or #d, where #v = variable and #d = value.                                                                                                      
  If you use #v in the format, you have to enter the variable names when you define prompts for the module.  
  If you use only #d in the format, you do not have to enter the variable names when you define prompts. The variables are passed to the program in the order they are created, and the program reads them sequentially and identifies them by their order in the command. |
| Command path  | The subdirectory where programs of this type are stored. This field works in conjunction with the library defined for the module. If you leave the field blank, AppWorx will expect to find the program in the directory specified in the Library field. |
| File extension| The default extension on the program file name. For example, .sql for SQL*Plus programs. This field is required if the programs have an extension.                                                                                               |
8.4 Entering General Information for Modules

8.4.5 Commonly Used Program Types That Ship with AppWorx

Below are several examples of commonly used AppWorx program types:

- BOURNE: Runs a Bourne or Korn shell script.
- COMMANDS: Runs a Bourne or Korn shell script and can register output.
- EXEC: Runs a Bourne or Korn shell script and can register output, and scripts must reside in the EXEC subdirectory.
- LOADP: SQL load, used for loading a flat text file into a database.
- SHELLS: Runs a Bourne or Korn shell script and cannot register output, and scripts must reside in the EXEC subdirectory.
- SQLP: Runs SQL Plus scripts.
- SURUNP: Runs scripts under a different user’s login.

AppWorx also uses program types to run jobs for various applications including Oracle Applications, Banner and SAP. For descriptions of additional program types and their scripts, see the AppWorx Technical Reference. If you need to create a new program type and matching program type script, call AppWorx Technical Support at 1-877-APPWORX for assistance.
8.4 Entering General Information for Modules
8.4.6 Entering Execution Options

The execution options determine what happens when a module runs. You set the execution options on the General tab of the Modules window.

You can set a variety of options that determine what happens when a module executes in the Execution options box (shown in Figure A) on the General tab of the Modules window.

![Figure A. The Execution options box, found on the General tab of the Modules window.](image)

The Execution option fields are described in the table below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>The default AppWorx queue the job will be submitted to. If the module is included in a chain, the chain queue overrides this setting. If the module is submitted on an ad hoc basis, the user may have the option to change the queue setting in the Submit window.</td>
</tr>
<tr>
<td>Priority</td>
<td>Determines when a module is in relation to other modules in the same queue. A module with a lower priority number will be run before modules with higher numbers. The default priority is 50. Note: Modules submitted with 0 priority will have the status of NO PRIORITY in the Backlog and will not run.</td>
</tr>
<tr>
<td>Max run time</td>
<td>Used to prevent runaway programs. It determines how long the program can run before timing out (DDD:HH:MI). The default time is 00:00:00, which lets the job run forever. If a job times out, it is given a status of TIMEOUT.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
Min run time | Used to flag jobs that run in a shorter time than expected (DDD:HH:MI). The default time is 00:00:00 which effectively inactivates this option. If a job completes in less than the minimum run time, AppWorx assigns the job a status of ABORTED.

Ave run time | Place to store the average run time for a job (DDD:HH:MI). This time will be displayed in the Gantt chart for Chain Summaries, Graphical Forecasts, and Chain Flows. For more information on Chain Summaries, see topic 6.2 Viewing Chain Summaries. For more information on Chain Flows, see topic 9.6 Simulating Chain Flow Execution.

Single run | When selected, two instances of the module cannot run concurrently.

Keep history records | When selected, the system will store the system output from the module. The output will be stored for the number of days set in the Retention days field on the module’s Output tab. The output is useful for debugging jobs that have aborted. If this option is not selected, AppWorx will not display a record of the job in the History as long as the job completes successfully. If the job aborts, a record will be entered in the History.

Stay in queue on abort | When selected, the aborted module remains in the Backlog so it can be examined, repaired, and reset. The queue is not blocked by this aborted module, even if the queue has only one thread. If this option is not selected and the module aborts, it is moved to the History. The module may be examined from the History to determine why it aborted, but because there is no longer a record in Backlog, it cannot be repaired or reset. When this option is selected and the Restart once on abort option is not, and the module aborts you will see the original Job ID `<jobid>` in the History and the current listing `<jobid>.01` in the Backlog.

Restart once on abort | When selected, AppWorx will automatically restart a module the first time it aborts, but will not restart it if it aborts a second time. When this option and the Stay in queue on abort option are both set, and the module aborts, you will see three records for it in the Explorer window.
- The original listing for the Job ID `<jobid>` in the History shows that the module ran.
- A second listing `<jobid>.01` in History shows that it aborted.
- The current listing `<jobid>.02` in the Backlog for the restarted module.
8.5 Specifying Output Options for Modules

Using the Output tab, you can control what happens to the output of modules.

Using the output tab shown in Figure A, you can control what happens to the output of modules.

Procedure

To set the module output options:

1. From the output options on the Output tab, complete the fields using the information in Table A.
2. If you wish, you can select the other tabs on the Modules window to edit them at this time.
3. Click Apply to add the module definition to the list of modules, or OK to add the new definition to the list of modules and close the Modules window.

Table A. Module Output tab descriptions

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printer group</td>
<td>The generic type of device that the job will be sent to (for example: LASER_PRINTER, FAX, <a href="mailto:PBROWN@APPWORX.COM">PBROWN@APPWORX.COM</a>, TRAINING_DEPARTMENT, or BELLEVUE). The printer group you select determines the devices displayed in the Printer Name list box.</td>
</tr>
<tr>
<td>Printer</td>
<td>Specifies the default output device or set of devices where the output will be sent to (for example: HPLJ5_Ground, Accounting Laser, Atlanta Laser). You must select a printer group before you can select a printer. <strong>Note:</strong> Printer settings in the module definition can be overridden settings in a chain.</td>
</tr>
<tr>
<td>Printer option</td>
<td>Used for specifying dynamic printer options for the selected printer. This value or list of values is defined by the spooler assigned to the printer.</td>
</tr>
<tr>
<td>Output function</td>
<td>Determines how output is handled. There are three choices: LOG: The output is available for viewing from the Jobs/Explorer and the Output windows. PRINT: The output is printed and is available for viewing from the History and the Output window. To see the output on the Output window, a user must query for jobs with a PRINT status. STORE: The output is stored and available for viewing from the History and the Output window. To see the output on the Output window, the user must query for jobs with an STORE status.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Default copies</td>
<td>Sets the default number of copies to be printed.</td>
</tr>
<tr>
<td>Retention days</td>
<td>Sets the number of days the report output and system output generated by the program will be stored on the system before it is deleted. The range is from 0 to 9999 days. If set to zero, the current day is not counted, and the output of the module will be deleted the next day.</td>
</tr>
<tr>
<td>Max # of revisions</td>
<td>Determines how many past versions of a module’s output will be stored on the system. When you select <strong>Max # of revisions</strong>, AppWorx finds the oldest revisions and deletes them so that only the maximum number of output entries (revisions) are left. If a program generates multiple reports as separate files, each file counts against this setting. For example, if a program produces two output files, and you want to retain the last three versions of each of them, you would set <strong>Max # of revisions</strong> to 6 (2 output files x 3 revisions = 6). AppWorx treats system output as one group, deleting all output assigned to the SYSOUT printer for a particular module up to the maximum number of revisions. Then it deletes all outputs up to the maximum number of revisions for all the other printers. When it looks at all output for a particular type of printer and a specific module, it uses the alias for the module. Therefore, AppWorx deletes the maximum number of revisions for each module for the system output printers, and the maximum number of revisions for each module for every other printer. <strong>Note:</strong> This option is not active unless the <strong>Use module max revisions</strong> option is turned on for the agent. For more information on user options, see topic 17.2 Defining Agents.</td>
</tr>
<tr>
<td>Program options</td>
<td>Passes standard command line options to the program called by the module. The options should be those that could not be defined in the program type. The value of this field is set in the operating system environment variable <strong>rpf_options</strong>.</td>
</tr>
<tr>
<td>Output directory</td>
<td>AppWorx normally sends output to the directory specified for the agent. Leave this field blank unless the program requires the output for the module (but not the system output) to be sent to a different directory. <strong>Note:</strong> You can use substitution variables in this field. This setting is overridden if a directory is entered in the <strong>Output Directory</strong> field on the Users window. For more information, see topic 16.2 Defining Users.</td>
</tr>
<tr>
<td>Print system output</td>
<td>When selected, the system output from the module is tagged for printing. By running the script <strong>$AW_HOME/sql/sysout.sql</strong>, you can print the system outputs of all modules that have been checked. To print these system outputs, create a module to run <strong>sysout.sql</strong>. <strong>Note:</strong> The module that runs the <strong>sysout.sql</strong> script must have a printer assigned to it in order for anything to be printed.</td>
</tr>
<tr>
<td>logs</td>
<td>Abort if no output When selected, the module will be considered aborted if no output is generated from the module. Use this option for modules whose key function is to produce a printed report.</td>
</tr>
</tbody>
</table>
8.6 Setting Module Logins

Using the Login tab, you can set the login the system will use when the module is executed.

Using the Login tab for modules, you can set the login the system will use when the module is executed (if necessary).

Procedure

To set the login options:

1. From the Login tab, complete the fields using the information in Table A.
2. If you wish, you can select the other tabs on the Modules window to edit them at this time.
3. Click Apply to add the module definition to the list of modules, or OK to add the new definition to the list of modules and close the Modules window.
Table A. Module Output options descriptions

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login ID</td>
<td>If the program run by a module requires access to a database or server, you can set the login the system will use when the module is executed. This setting can be overridden for modules that are included in a chain when a login is selected in the Login field on the Components tab. For information on assigning logins to chain components, see topic 9.4.2 Setting Component Execution Options.</td>
</tr>
</tbody>
</table>
| Login source options | These options control the login the system will use when the module is executed.  
  - Default: Uses the option selected for the master’s local agent. For more information, see topic 17.2 Defining Agents.  
  - Module: Uses the login assigned in the Login ID field.  
  - User's: Uses the login assigned to the user (if one is assigned). This option is useful if you want to run a report based on the user’s view of a database. For more information, see topic 16.2 Defining Users. |
8.7 Adding Notes

You can enter customizable general and abort notes for modules, chains, and chain components. By selecting an individual job and choosing Notes, operators can access these comments, suggestions, or instructions.

Notes provide a location to enter relevant information about the processing of jobs. Notes for a sample FTP module are shown in Figure A.

![Figure A. The Notes tab for an FTP module.](image)

**Types of Notes**

There are two types of customizable notes: General and Abort categories. General notes can contain information on goals and requirements or existing security and access issues. Abort notes can contain information on what action to take if a module aborts or fails, who to contact if a module aborts, or what considerations exist when running a job.

**Note Locations**

There are three locations within AppWorx where you can enter notes. Suggested uses for notes are described in the table below:

<table>
<thead>
<tr>
<th>Note type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>Useful when the module is going to be requested ad hoc, or when the notes for the module would be useful—regardless of how the module was invoked.</td>
</tr>
<tr>
<td>Chain</td>
<td>Can be used to provide information which is relevant to the entire chain.</td>
</tr>
<tr>
<td>Chain component</td>
<td>Can be used to provide specific information about one component in a chain.</td>
</tr>
</tbody>
</table>
Procedure

To add notes to a module, chain, and chain component:

1. From the Notes tab of the module, chain, or chain component, select an option from the Note type drop-down box.
2. Type in comments, tips, suggestions, and precautions.
   When entering notes remember:
   • Lines do not wrap. To start a new line, press Enter.
   • You can cut and paste text.
3. If you wish, you can select the other tabs on the Modules window to edit them at this time.
4. Click Apply to add the module definition to the list of modules, or OK to add the new definition to the list of modules and close the Modules window.

Viewing Job Notes

Users can view module, chain, or chain component notes from the Backlog and History. If a job aborts, an operator can view the specific abort notes and make more effective operational decisions. For more information, see topic 5.9.4 Viewing Job Notes.
8.8 Assigning Roles to Modules

Select the Roles tab and use the arrow keys to assign roles to a module.

Roles are used in AppWorx control access to objects. Roles can be added to a module from the Roles tab as well as from the role itself.

If you are assigned to a Maintenance role, all modules you create will automatically be assigned to your role when you define them. You can view a list of objects that you have access to from the View menu by selecting View Assigned Objects. If you do not have access to the object, see your AppWorx administrator.

**Procedure**

To assign roles to a module:

1. From the Roles tab, assign and unassign roles by double-clicking on the role names. You can also highlight a role and select the arrow buttons to assign and unassign them. Use the double arrow buttons to assign or unassign all roles to the module.

2. Select a module/chain option for each role in the Usage column. The module/chain options are described below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both</td>
<td>Users can submit these modules and view the output with the file viewer.</td>
</tr>
<tr>
<td>Output only</td>
<td>Users can view the output of these modules, but not request them.</td>
</tr>
<tr>
<td>Request only</td>
<td>Users can submit these modules, but not view their output.</td>
</tr>
</tbody>
</table>
3. If you wish, you can select the other tabs on the Modules window to edit them at this time.

4. Click **Apply** to add the module definition to the list of modules, or **OK** to add the new definition to the list of modules and close the Modules window.

For more information on assigning roles, see chapter 16: *AppWorx Security*. 
8.9 Copying Modules

To copy a module, choose Copy on the Modules selector window. You can decide whether to copy the conditions, notes, and schedules for the module.

There may be times when you want to create several modules that are similar except for a few minor changes. For example, you may want to create two modules that run two different programs, but the information for the programs is identical except for the program names. The Copy feature lets you create one module, then copy it and change the program name.

When you copy a module, the prompts also are copied. You have the option of copying its conditions, notes, and schedules.

**Figure A.** To copy a module, highlight a module and select the Copy button.

**Prerequisites**

There are several prerequisites to keep in mind when copying a module:

- You must have edit access to the module. This access is controlled by roles. If you created the original chain, you probably have edit access to it.
- The module you want to copy must be displayed on the Modules selector window.

If you are copying a module recently created by another user, you may need to refresh your objects by going to the View menu and selecting Refresh Assigned Objects.
Procedure

To copy a module:

1. From the Modules selector window, highlight the module you wish to copy.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.
2. Click Copy.
   AppWorx displays the Copy window shown in Figure A.
3. Enter the name and description for the new module.
4. Select the appropriate check boxes to copy conditions, notes, and schedules.
5. To accept the information and close the window, click OK.
   AppWorx closes the Copy window, adds the module to the Modules selector window and opens the new module’s definition window.
# 9 Creating Chains

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1 Introduction to Creating Chains</td>
<td>190</td>
</tr>
<tr>
<td>9.2 Defining Chains</td>
<td>192</td>
</tr>
<tr>
<td>9.3 Adding Chain Components</td>
<td>194</td>
</tr>
<tr>
<td>9.4 Updating Component Definitions</td>
<td>196</td>
</tr>
<tr>
<td>9.4.1 Setting Component Schedules</td>
<td>198</td>
</tr>
<tr>
<td>9.4.2 Setting Component Execution Options</td>
<td>200</td>
</tr>
<tr>
<td>9.4.3 Specifying Component Output Devices</td>
<td>202</td>
</tr>
<tr>
<td>9.4.4 Specifying Component Prompt Values</td>
<td>204</td>
</tr>
<tr>
<td>9.4.5 Specifying Component Predecessor Statements</td>
<td>206</td>
</tr>
<tr>
<td>9.4.6 Specifying Component Conditions</td>
<td>208</td>
</tr>
<tr>
<td>9.5 Viewing Chain Flows</td>
<td>210</td>
</tr>
<tr>
<td>9.6 Simulating Chain Flow Execution</td>
<td>212</td>
</tr>
<tr>
<td>9.7 Copying Chains</td>
<td>214</td>
</tr>
<tr>
<td>9.8 How AppWorx 5.1 Threads Chain Components</td>
<td>216</td>
</tr>
<tr>
<td>9.8.1 Example: How AppWorx 5.1 Threads Chain Components</td>
<td>218</td>
</tr>
<tr>
<td>9.9 Why Use a Schedule Module?</td>
<td>220</td>
</tr>
<tr>
<td>9.10 Defining Schedule Modules</td>
<td>222</td>
</tr>
</tbody>
</table>
9.1 Introduction to Creating Chains

Chains are used to schedule and execute one or more modules and other chains. Chains transfer the chore of routine maintenance and scheduling from operations personnel to AppWorx.

In many operations environments, shell scripts are used to run programs. A shell script or job stream may run a series of programs in a specific order, on certain days, and under certain conditions. In AppWorx, chains take the place of shell scripts or job streams. A chain includes one or more chain components (modules and chains assigned to the chain), general scheduling information for the chain, specific scheduling information for each chain component, and conditions that must be met for each component to run. Since a component can be either a module or chain, AppWorx gives you a great deal of flexibility in building chains. A chain is shown in Figure A.

Note: AppWorx roles control access to chains. If you do not have access to them, see your AppWorx administrator.

Running Chains

There are two ways to run a chain:

- Enter scheduling information using the Schedule tab on the Chains window.
- Submit the chain by opening to the Operations menu and selecting Requests.

In prior versions of AppWorx, schedule modules were needed to submit any chain on an ad hoc basis. With version 5.1, chains are included in the Requests window. For more information, see topic 9.9 Why Use a Schedule Module?

999 Components and 32 Levels in a Chain

A chain may contain up to 999 components. A chain can include a chain, which in turn includes another chain. This is called nesting or sub-chaining. You can sub-chain to 32 levels. As a rough guide, 4 levels of sub-chaining is sufficient for most implementations.
Using Chain Components as Pointers

Chain components are modules and chains assigned to the chain. When you add a component to a chain, AppWorx creates a pointer from the chain to the module or chain instead of adding a copy of it. This is in keeping with the AppWorx object-oriented approach. If you edit a module or chain definition, the update is effective immediately for all chains that include it.

Steps for Creating a Chain

The basic steps for creating a chain are:
1. Create the chain object.
2. Set the general and execution options.
3. Add scheduling information to the chain.
4. Add components to a chain.
5. Schedule the components within the chain.

Adding Notes to Chains and Chain Components

Notes provide a location to enter relevant information about the processing of a job. By selecting an individual job and choosing Notes, operators can access these comments, suggestions, or instructions with respect to an individual job. Notes include an Abort and a General category text editor.

You can enter customizable general and abort notes for modules, chains, and chain components. For more information, see topic 8.7 Adding Notes.
9.2 Defining Chains

To create a new chain, complete the required fields on the General tab. The execution options determine what happens when a component runs. You set the execution options on the General tab of the Chains window.

The minimum requirement for creating a chain is filling in the required fields on the General tab. However, the chain cannot do any ‘work’ until you add components to it.

You can define certain objects by clicking the icon to the right of their field. AppWorx will display an edit window where you can define the new object. Fields marked with an ‘*’ on their right are required.

Procedure

To add an AppWorx chain:

1. From the Chains selector window, click New.
   AppWorx opens the Chains window shown in Figure A.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Enter a name and description for the chain on the General tab of the Chains window shown in Figure A.
   The entries may be up to 30 characters long each.

3. Complete the fields using the information in Table A.

4. If you wish, you can select the other tabs on the Chains window to edit them at this time.

5. To add the chain definition to the list of chains, click Apply. Or to add the new definition to the list of chains and close the Chains window click OK.
   AppWorx displays an error message to alert you to any unspecified fields when they need to be filled in.
### Table A. The fields for the Chains General tab:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Applications are the general category to which modules and chains belong (for example: inventory, accounts payable, etc.). They are used on the Requests window to limit the list of modules and chains. For information on defining applications, see topic 8.4.2 Defining Applications.</td>
</tr>
<tr>
<td>Agent/Group</td>
<td>The agent or agent group where the program you wish to run is stored and where it will run from. If “No selection” is picked, chain components will run on the agent selected on the module’s General tab.</td>
</tr>
<tr>
<td>Queue</td>
<td>The default AppWorx queue the chain will be submitted to. If a module is included in the chain, the chain queue overrides the module queue. If the chain is submitted on an ad hoc basis, the user may have the option to change the queue on the Submit window.</td>
</tr>
<tr>
<td>Priority</td>
<td>Determines when a chain is run in relation to other modules and chains in the same queue. A module or chain with a lower priority number will be run before modules or chains with higher numbers. The default priority is 50. <strong>Note:</strong> Modules submitted with 0 priority will have the status of NO PRIORITY in the Backlog and will not run.</td>
</tr>
<tr>
<td>Ave run time</td>
<td>Place to store the average run time for a job (DDD:HH:MI). This time will be displayed in the Gantt chart for Chain Summaries, Graphical Forecasts, and Chain Flows. For more information on Chain Summaries, see topic 6.2 Viewing Chain Summaries. For more information on Chain Flows, see topic 9.6 Simulating Chain Flow Execution.</td>
</tr>
<tr>
<td>Single run</td>
<td>When selected, two instances of the chain cannot run concurrently.</td>
</tr>
<tr>
<td>Keep history records</td>
<td>When selected, the system will store the system output from the chain for the number of days defined in the HISTORY_PURGE module. The output is useful for debugging jobs that have aborted. If this option is not selected, AppWorx will not display a record that the chain ran in the History as long as it completes successfully. If the chain aborts, a record will be entered in the History.</td>
</tr>
</tbody>
</table>
9.3 Adding Chain Components

Chain components identify the programs that will be run as part of a chain. You can add both modules and chains to a chain. You can add up to 127 chain components to a chain.

For a chain to accomplish work, you must assign one or more components to it. Components can include both modules and chains. You can add up to 127 components to a chain. Figure A shows a chain with five components.

The Components tab shown in Figure A includes three panes:

- Left: the modules and chains that are available for addition to the chain.
- Center: the sub-tabs that define each components.
- Right: the chain’s components.

Each of the panes can be expanded by dragging the splitter bars, or clicking their arrows. When you define a new chain, you will be adding modules and/or chains—so all three panes are displayed. When you edit an existing chain, you will most likely not need to add additional modules/chains—so the left pane is collapsed by default.

When you add one or more components, a triangle is displayed on the Components tab to signify unsaved changes. The triangle is also displayed next to the Components label on the right side of the screen. For more information on updating unsaved changes, see topic 8.3 Updating Unsaved Changes.
Example Chain

The chain shown in Figure A is set up to transfer a file, load the data in the file, and run reports. The components assigned to the chain run programs that perform the following actions:

• The FTP_ORDERS module transfers the file.
• The LOAD_DATA module loads the data into a database table.
• The remaining modules run reports against the database.

Procedure

To add one or more modules and/or chains to a chain:

1. Select a module or chain from the list in the Modules/Chains box on the left side of the Components tab on the Chains window shown in Figure A.
   AppWorx displays the General sub-tab for the module or chain in the center pane of the screen.
   You can filter the list by selecting an application. Applications specify a group where modules are assigned. Only the applications modules and chains assigned to you via roles will be displayed.
   You can type the first few letters of a module or chain’s name in the Search field, and AppWorx will highlight it. If two modules or chains start with the letter you type, AppWorx highlights the first one.
2. If you wish, you can edit the fields on any of the tabs in the center pane for the module or chain, to affect the instance of the module or chain.
   Each of the tabs will be described in the following topics. You can edit these tabs later.
3. Click Add.
   AppWorx adds the component to the chain and puts the Search box into focus, so you can add more components.

If you wish to reorder the components, use the arrow buttons on the right side of the screen.

Adding Notes to Chains and Chain Components

Notes provide a location to enter relevant information about the processing of a job. By selecting an individual job and choosing Notes, operators can access these comments, suggestions, or instructions with respect to an individual job. Notes include an Abort and a General category text editor.

You can enter customizable general and abort notes for modules, chains, and chain components. For more information, see topic 8.7 Adding Notes.
9.4 Updating Component Definitions

After creating a chain, you can set execution options for each component, select its eligible run days, and define special run conditions.

After adding components to a chain, you can use the execution options to control how, when, and where the component executes.

From the General sub-tab shown in Figure A, you can:

- Set aliases for the components.
- Choose to propagate alias names in predecessors statements.
- Activate/inactivate components.
- Refine component schedules.
- Set component execution options.

If you make changes on any of the Component tab’s sub-tabs, a red triangle is displayed on the Components tab to signify unsaved changes. The red triangle is also displayed next to the Components label on the right side of the screen. For more information on updating unsaved changes, see topic 8.3 Updating Unsaved Changes.
Setting Aliases for Chain Components

The Alias field lets you give an alternate name to a chain component. It gives you a way to distinguish multiple instances of the same module or chain added to a chain. For example, you might use a module twice in a chain, each with a different set of values entered for the prompts. You can use the Alias field to give each module a unique name. Aliases replace the module/chain name in the Backlog and History. The default alias for a component is its module or chain name.

Propagating Alias Names

If you change a component’s alias, you can propagate its references in any predecessor statements by selecting the Propagate Alias box. When this option is selected and you change a component’s alias, AppWorx will update the component’s reference in all predecessor where it is referenced. For information on using predecessor statements, see chapter 12: Assigning Predecessor Statements.

Activating and Inactivating Chain Components

You can set the status of a chain component to Active or Inactive using the radio buttons in the upper right corner of the pane. This can be useful if you are building a chain and want to test some modules or chains but not others. An inactive module/chain’s status will appear as ‘x’ in the thread column of the Components list.

Subtopics

Several scheduling subtopics follow:

9.4.1 Setting Component Schedules
9.4.2 Setting Component Execution Options
9.4.3 Specifying Component Output Devices
9.4.4 Specifying Component Prompt Values
9.4.5 Specifying Component Predecessor Statements
9.4.6 Specifying Component Conditions
9.4 Updating Component Definitions

9.4.1 Setting Component Schedules

You can set the days of the week a component will run within the limits set for the chain schedule. You can also choose to run on or skip the days specified in a calendar.

If you are building a chain with a number of different components, you may want a component to run only on certain days of the week even though the chain runs every day. Or you may want a component to run on or skip the days specified in a calendar. You can do both with the Schedule box on the General sub-tab for a chain component.

![Figure A. You can control the days a component runs.](image)

Setting the Days

To set the days a component will run, select the days of the week. The days of the week selected for the component must be included in the days of the week selected for the chain. If the chain does not run on a specific day, components in the chain cannot run.
Using a Calendar

To use a calendar:

1. From the General sub-tab, select a calendar from the Calendar drop-down list box.
2. Choose an option:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Run    | Runs the component on the days specified in the calendar provided the calendar days:
|        | • Fall on one of the days of the week selected for the component.
|        | • Are within the administrative start and end dates set for the chain. |
| Skip   | Prevents the component from running on the days specified in the calendar. |

3. Select the days of the week if you are using the Run option.

If the appropriate days of the week are not selected, the component will not run even if the dates are included in a run calendar.
9.4 Updating Component Definitions

9.4.2 Setting Component Execution Options

Use the execution options to control how and where the component executes.

You can define the execution options for each chain component’s General sub-tab. The execution options are described in Table A.

Table A. Execution options

<table>
<thead>
<tr>
<th>Execution Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>If the chain component is assigned to an agent group, you can select an agent from the group (only the agents in the group will be listed). This job will run on the agent assigned here. If “No selection” is picked, the job will run on the agent selected on the chain’s General tab. If “No selection” is picked there as well, the job will run on the agent selected on the module’s General tab.</td>
</tr>
<tr>
<td>Login</td>
<td>Designates the login to be used to access the server or database required by the program. This overrides the login set for the module.</td>
</tr>
</tbody>
</table>
### Execution Option

<table>
<thead>
<tr>
<th>Execution Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Link**         | Determines how a component affects its chain in the Backlog:  
|                  | • Block on abort: Multi-threaded components immediately following this component will complete executing if it aborts. However, all other remaining components in the chain will not execute until the aborted job is cleared.  
|                  | • Unlink on abort: All remaining components will execute if this component aborts.  
|                  | • Unlink on start: Once this component launched, it will not affect the execution of the other components in the chain. |
| **Ave run time** | Place to store the average run time for a job (DDD:HH:MI). This time will be displayed in the Gantt chart for Chain Summaries, Graphical Forecasts, and Chain Flows. For more information on Chain Summaries, see topic 6.2 Viewing Chain Summaries. For more information on Chain Flows, see topic 9.6 Simulating Chain Flow Execution. |
| **Use request printer** | This option is only relevant for components belonging to chains which users will submit on an ad hoc basis. When selected, AppWorx uses the printer and settings selected on the Submit window rather than on the Output sub-tab explained in topic 9.4.3 Specifying Component Output Devices. |
| **Use module conditions** | If this option is selected, the component will use the conditions defined in the module or chain's definition. |
| **Multi thread** | Used to set the chain component to run by itself or simultaneously with other components.  
|                  | With this setting a component is multi-threaded, it will start when all earlier single-threaded components have completed executing. If there are several multi-threaded components grouped together in a chain, they will all be eligible to run at the same time.  
|                  | Without this setting a component is single threaded, it will not start until all earlier components in the chain have completed executing. For example, if the first component in a chain is a single-threaded module and it is put on hold, the other components in the chain will not execute. They will show a status of THREAD WAIT. The other components will not execute until the module is taken off hold and completes executing, or is deleted. This will be true even if the chain is submitted to a queue that has more than one thread. |
9.4 Updating Component Definitions

9.4.3 Specifying Component Output Devices

You can specify printers and output options for a chain component on the Output sub-tab.

You can specify a target printer or printers for the output of a component from the Output sub-tab. This setting overrides the general printer selection for the module. In turn, this setting can be overridden if the **Use request printer** option on the Chain components’ General sub-tab. Output will be sent to all printers listed on the Chain components’ Output sub-tab. The output options are described in Table A.

*Figure A.* To specify a printer for a component, click the Output sub-tab.
### Table A. Output options

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printer</td>
<td>Name of the Printer.</td>
</tr>
<tr>
<td>Option</td>
<td>Allows users to request a specific variable, setting, address, or orientation. This field appears only when a printer that uses the variable print option has been defined for the selected printer.</td>
</tr>
<tr>
<td>Function</td>
<td>There are three choices for directing the output of a component: PRINT: The output prints to the designated printer and the output file appears on the Output window. LOG: The output file appears on the Output window. STORE: The output file is stored on the Output window and is not viewable. To view the output, the user must query for the stored item on the Output window.</td>
</tr>
<tr>
<td>User</td>
<td>All user names are available for this field. The name you select will be displayed in the banner of reports if the -t %USR% option was specified in the Title field in the Spooler window.</td>
</tr>
<tr>
<td>Copies</td>
<td>Specifies the number of copies to print if sending to printer (1-99).</td>
</tr>
<tr>
<td>Available on</td>
<td>Select the days you want the printer to be available. This can be useful if you want the output generated by the component to be sent to one group of people on one day, and to another group of people on another day. For example, you might want a sales report sent to an area office printer every day, but to a regional office printer only on Fridays.</td>
</tr>
</tbody>
</table>
9.4 Updating Component Definitions

9.4.4 Specifying Component Prompt Values

To specify values for prompts in a chain component, click the Prompts sub-tab.

When you run a module or chain on an ad hoc basis, you can enter values for the prompts in the Submit window. For the chain components, you must enter the values for the prompts from the Prompts sub-tab. Different prompt values can be entered for each schedule.

The values for a prompt can be entered directly, or selected from lists. A list will be available if the data type selected for the prompt includes a SQL statement. You will be able to select a single value if it is a List-of-Values (LOV) prompt, or multiple values if it is a multi select (MS) prompt.

This topic presents a brief discussion of prompts. For a detailed discussion of prompts, see chapter 11: Adding Prompts to Modules and Chains.

Procedure

To specify prompt values for a chain component, enter a value in the Value column for each prompt on the Prompts sub-tab as shown in Figure A.

If the prompt uses a data type that includes an appropriate SQL statement in its definition, the LOV button will be active. Click it to open the List of Values window (see Figure B). Double-click a value to add it to the Value column.
If the prompt uses a data type that includes an appropriate SQL statement in its definition, the **MS** button will be active. Click it to open the Multiple Selection Values window (see Figure C), and select one or more values.

![Figure B. The List of Values window.](image)

**Figure B. The List of Values window.**

![Figure C. The Multi Selection Values window.](image)

**Figure C. The Multi Selection Values window.**
You can specify conditions for chain components.

You can define one or more predecessors that must be met in order for a chain component to run. If you do not specify conditions for a component, it will run on the days that it is scheduled (unless a condition prohibits its execution).

Procedure

To add a predecessor statement to chain component:

1. From the Predecessors sub-tab, select the AND or OR option.
   - To require both this statement and the previous predecessor statement to be true for this task to run, select AND.
   - To allow either this statement or the previous predecessor statement to be true for this task to run, select OR.

For the first predecessor statement, disregard the AND and OR radio buttons.
2. Select the module or chain that must have succeeded, failed, or completed since the last virtual workday start time.

You can select a module or chain from the Modules/Chains sub-tab, or a chain component from the Chain Components sub-tab.

To select a chain component, you must first select its chain (PRODSCH is selected in Figure B)

Check the Alias box to display the chain component’s name (or alias if defined), when it is added. By not checking this box, AppWorx displays the chain component’s order in the chain instead.

If you select the Alias box for a module with an alias, it is recommended that you check the Propagate Alias box for the chain component. For more information on the Propagate Alias box, see topic 9.4 Updating Component Definitions.

3. Select an option from the statement’s drop-down box.

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Run this task if the selected module/chain ran successfully.</td>
</tr>
<tr>
<td>Success (Skip on Failure)</td>
<td>Run this task if the selected module/chain ran successfully. Skip this task if the module/chain failed with a status such as DIED, ABORTED, or TIMEDOUT.</td>
</tr>
<tr>
<td>Failure</td>
<td>Run this task if the selected module/chain ran and failed with a status such as DIED, ABORTED, or TIMEDOUT.</td>
</tr>
<tr>
<td>Failure (Skip on Success)</td>
<td>Run this task if the selected module/chain ran and failed with a status such as DIED, ABORTED, or TIMEDOUT. Skip this task if the module/chain ran successfully.</td>
</tr>
<tr>
<td>Complete</td>
<td>Run this task if the selected module/chain ran and completed, failed, or was deleted or skipped.</td>
</tr>
</tbody>
</table>

4. Click Add.

AppWorx adds the predecessor statement to the box above. When the job is run, all predecessors must be met before it can execute. Jobs that are waiting for predecessors to be met display PRED WAIT as their status.
You can specify conditions for chain components.

You can define one or more conditions, that if met will prohibit the component from running. If you do not specify conditions for a component, it will run on the days that it is scheduled (unless a predecessor statement prohibits its execution).

This topic provides a quick introduction to conditions. For a more detailed discussion of conditions, see chapter 13: Working with Conditions.
Procedure

To specify conditions for a component:

1. To add a new condition, click **New** on the Conditions sub-tab, shown in Figure A. AppWorx displays the Condition Details window shown in Figure B.

![Figure B. The Condition Details window](image)

2. Select the timing when the conditions should be checked: BEFORE, DURING, DELETED, or AFTER.

3. Select a Condition value and Test.
   For details on completing these fields, see chapter 13: *Working with Conditions.*

4. Determine if the action will be taken the first time the condition is true, every time the condition is true, or if the condition should be disabled.

5. To accept the settings, click **OK.**
9.5 Viewing Chain Flows

You can view a chain flow to see how a chain’s components will execute by clicking the Chain flow button.

Chain flows allow you to graphically view a chain’s execution. To use the Chain flow feature, you must have the Graphical Analysis Package feature. For more information, see topic 1.6 AppWorx License Options.

To view a chain flow from the Chains window, click the Chain flow button in the bottom left corner (see Figure A).

AppWorx displays the Chain flow window shown in Figure B. It has three panes which include:

- A tree of all scheduled modules and chains.
- A table with the details of each module/chain. The table includes a Gantt chart used to show estimated run times of chains and their components.
- A graphical display of all scheduled modules and chains. Modules are displayed in rounded boxes. Chains are displayed as expandable boxes. The chains’ title bars include ‘+’ or ‘-’ buttons to show whether they are expanded or collapsed. Nested chains are shaded to differentiate their levels.

Selecting a component in one pane brings it into focus in both. In Figure A the LEVEL_3 chain is selected, notice it is highlighted in the tree and has a border around it in the flow.

You can maximize the Chain Flow window and the AppWorx desktop by going to the File menu and selecting Screen to full size.
Resizing the Display

AppWorx automatically resizes the display when you expand or collapse a chain. You can also resize the flowchart’s display in the following ways:

- Expand the viewable area with the splitter bars, by dragging them or clicking their arrows.
- Use the 0-200 percentage scale to manually adjust the scale.
- From the flowchart’s View menu, select from the following boxes to automatically adjust the display:
  - Fit to size
  - One time fit to size
  - Show vertical (orients flowchart vertically)
  - Show horizontal (orients flowchart horizontally)

Display Options

From the flow chart’s View menu, you can use the check boxes to choose whether to have AppWorx display:

- Predecessor relationships and execution order (using the Show Predecessors, Show Order, Show all arrows, and Show no arrows options).
- Only the relationships to the module/chain selected with the mouse (using the Draw to/from dependencies only option).

Next Step

Now you have opened the flow window. The next step is to simulate the chain’s execution. For more information, see topic 9.6 Simulating Chain Flow Execution.
9.6 Simulating Chain Flow Execution

You can view a simulation of a chain flow by going to the Simulation menu, selecting options, and selecting Start.

Procedure

To view a simulation of a selected chain’s execution:

1. Go to the Simulation menu and check or uncheck the following options:
   - Fast: Speed up the simulation.
   - Auto scroll: Scrolls to the active component in the simulation.

2. Select Start to run the simulation.

   AppWorx runs through the simulation, color coding the modules as it goes.

You can stop or pause the simulation at any time from the Simulation menu.
9.7 Copying Chains

To copy a chain, choose Copy on the Chains selector window. You can decide whether to copy the conditions, notes, and schedules for the chain.

There may by times when you want to create several chains that are similar except for a few minor changes. For example, you may create two chains with the same set of five modules, but add a sixth module to one of the chains. You can use the copy feature to do this quickly without having to recreate the entire chain.

When you copy a chain, you also copy its prompts. You decide whether to copy its conditions, notes, and schedules as well.

**Prerequisites**

There are several prerequisites to keep in mind when copying a chain:

- You must have edit access to the chain. This access is controlled by roles. If you created the original chain, you probably have edit access to it.
- The chain you want to copy must be displayed on the Chains selector window.

If you are copying a chain recently created by another user, you may need to refresh your objects by going to the View menu and selecting Refresh Assigned Objects.
Procedure

To copy a chain:

1. From the Chains selector window, highlight the chain you wish to copy.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.
2. Click **Copy**.
   AppWorx displays the Copy Chain window shown in Figure A.
3. Enter the name and description for the new chain.
4. Select the appropriate check boxes to copy conditions, notes, and schedules.
5. To accept the information and close the window, click **OK**.
   AppWorx closes the Copy window, adds the chain to the Chains selector window and opens the new chain's definition window.
9.8 How AppWorx 5.1 Threads Chain Components

All components called by a chain are threaded together. Modifying chains can affect the total chain. Chains will go into an INITIATED status once their components have been inserted into a queue.

There are several improvements made to AppWorx 5.1 for threading components in chains.

In AppWorx 5.1 these behaviors are implemented:

• Chains can be called by a chain.
  Schedule modules were required to call a chain in previous versions of AppWorx. They are only used in unusual circumstances. For more information, see topic 9.9 Why Use a Schedule Module?

• Chains go into an INITIATED status once their components have been inserted into a queue.

• The execution order of a chain, which includes one or more chains is evaluated the same as a chain with a only standard modules. When a chain is expanded in the Backlog, it is considered running until all its children have finished.

Note: The exception is an aborted chain component without ‘Block on abort’ selected in its Link field.
• Threading for chains is the same as chain components. This means that if two multi-threaded chains are invoked by the calling chain then all of their modules become eligible to be executed concurrently. However nested chains are not threaded with the calling chain.

• Chains stay in the Backlog until all their children are complete. Modifying these can affect the total chain. HOLD, KILL, DELETE, RESET will be attempted on each component and its children if the action is appropriate (considering the child’s current status).

• Chains (and schedule modules) in the Backlog do not count toward a queue’s thread limit.

• The elapsed time on a chain will be the total chain’s execution time.

Subtopic

The subtopic listed below provides an example of how AppWorx 5.x threads chains:

9.8.1 Example: How AppWorx 5.1 Threads Chain Components
9.8 How AppWorx 5.1 Threads Chain Components

9.8.1 Example: How AppWorx 5.1 Threads Chain Components

Below is an example of how AppWorx 5.1 threads chain components.

The following example components and chains in the diagrams in Figure A will be used to illustrate the improved chain behavior for AppWorx 5.1. They show the architecture and threading of a chain for AppWorx 4.x and 5.1. In these examples, assume that these components will be running in a queue assigned to an unlimited thread schedule.

**Threading in AppWorx 5.1**

AppWorx 5.1 (as illustrated on the left in Figure A) runs the example chains and components in this order:

1. **CHAIN_1** runs inserting modules A and F, and chains **CHAIN_2** and **CHAIN_3** into the Backlog.
   - **CHAIN_1** will stay in the Backlog with an INITIATED status, until all of these modules finish running.

2. **Module A**, and chains **CHAIN_2** and **CHAIN_3** run concurrently, because they are all multi-threaded.

3. When chain **CHAIN_2** runs it inserts modules B and C into the Backlog.
   - **CHAIN_2** remains in the Backlog with an INITIATED status until modules B and C complete.
   - Modules B and C are single threaded, therefore module C cannot run until module B finishes. However, other modules in the Backlog that are not a part of **CHAIN_2**, can run concurrently with either of these modules.

4. When **CHAIN_3** runs it inserts modules D and E into the Backlog.
   - **CHAIN_3** remains in the Backlog with an INITIATED status until modules D and E complete.

5. Module F will run as soon as module A, C, D, and E (and all of the modules run by their chains) are finished running.
Differences from Prior AppWorx Versions

Notice the following threading differences in Figure A.

- In AppWorx 5.1, chains are called by other chains without using schedule modules.

  **Note:** Schedule modules were required to call a chain through version 5.0. Now they are only used in unusual circumstances. For more information, see topic 9.9 Why Use a Schedule Module?

- The multi-threaded components in CHAIN_1 are called together.

*Figure A.* Chains can now be called by a chain. The components in these chains are nested in their own chain, and threaded independently from their parent chain.
9.9 Why Use a Schedule Module?

Schedule modules provide an alternative method of invoking a chain’s components. If you are new to AppWorx, you will probably never use schedule modules. They mostly exist in AppWorx 5.1 for legacy customers.

Schedule modules provide an alternative method of invoking a chain’s components. In prior versions of AppWorx you could not:

- Add chains directly to other chains.
- Submit chains on an ad hoc basis.

You had to create schedule modules to do these things. A sample schedule module is shown in Figure A.

In AppWorx 5.1, you can add chains to other chains and request chains from the Requests window. You can also add prompts to a chain. When you run a chain that includes prompts, the values from these prompts can be passed to the components in the chain using numeric substitution variables. With these improvements, it is no longer necessary to create a schedule module for your chains.

What Does a Schedule Module Do Now?

If you are new to AppWorx, you will probably never use schedule modules. They mostly exist in AppWorx 5.1 for 4.x and 5.0 customers. But you can still invoke a chain’s components with a schedule module instead of with the chain—if you want to call the chain in more than one way. With a schedule module you can include different prompts, predecessors, and conditions.
For example, you may choose to create a schedule module if you are passing prompt values to a chain’s components and you want to be able to request them with your choice from two sets of default values. One included in the chain’s definition, the other included in the schedule module’s.

To set this up you would do the following:

• In the Chains window, go to the Prompts tab (not to be confused with the Components’ Prompts sub-tab) and add the prompts that you wish to pass to the components.

• Add the same prompts to the schedule module, but with different values selected in the Default Value field.

• Add numeric substitution variables to the components in the chain as described in topic 11.10 Passing Values Through a Chain with Prompts.

If you submit the schedule module, any prompts associated with the chain are ignored.

**Note:** If you want to run a series of modules with two or more sets of default values without creating schedule modules, create two chains that include the same modules.

**Schedule Modules in the Backlog**

Schedule modules behave exactly like chains in the Backlog. When you request a schedule module, it goes into an INITIATED status once the chain components are inserted into a queue (see Figure B). Like a chain, you can right-click the schedule module to change its status, which in turn will affect the status of all eligible components.

For details on creating schedule modules, see topic 9.10 Defining Schedule Modules.
9.10 Defining Schedule Modules

To create a schedule module, go to the Modules selection window, click new and select a chain from the Select module type window.

Schedule modules provide an alternative method of invoking a chain’s components. Their uses are described in topic 9.9 Why Use a Schedule Module? A sample schedule module is shown in Figure A.

Figure A. Schedule modules provide an alternative method of invoking a chain's components.

Procedure

To create a schedule module:

1. Go to the Object Admin menu and select Modules, on the selector window and click New.
2. AppWorx Displays the Select module type window shown in Figure B.
3. Select Schedule, and choose the chain you wish to call.

AppWorx displays the Modules window shown in Figure A.

Notice the non-editable fields. Type displays SCHEDULE and Program name displays the chain name you selected.

4. Enter a name and description for the schedule module.

The entries may be up to 30 characters long each.
5. Select an application.
   Applications are the general category to which modules and chains belong (for example: inventory, accounts payable, etc.). For more information on defining applications, see topic 8.4.2 Defining Applications.

6. Select a queue.
   The chain will run in this queue instead of the queue assigned to the chain.

7. If you wish, you can fill in the non-required fields on the General and other tabs of the Modules window.

8. To add the module definition to the list of modules, click **Apply**. To add the new definition to the list of modules and close the Modules window, click **OK**.
   AppWorx displays an error message to alert you to any unspecified fields when they need to be filled in.

9. To test the schedule module, go to the Operations menu, select Requests and run the schedule module.
   **Note:** Schedule modules in the Backlog do not count toward a queue’s thread limit.
10

Scheduling Modules and Chains

10.1 Introduction to Scheduling Modules and Chains .................................................. 226
10.2 Entering General Information for Schedules ........................................................ 228
10.3 Entering Schedule Frequencies ........................................................................... 230
10.4 Entering Schedule Exceptions ............................................................................. 232
10.5 Specifying Prompt Values for Schedules ............................................................. 234
10.6 Defining Calendars .............................................................................................. 236
10.7 Example Schedules .............................................................................................. 238
10.8 How AppWorx Determines Eligibility ................................................................. 242
10.1 Introduction to Scheduling Modules and Chains

With AppWorx, you can create schedules to run modules and chains that account for days of the week, specific days of the month, and days in a calendar.

Adding Schedules

To define a new schedule for a module or chain:

1. Select the Schedule tab for a module or chain as shown in Figure A.
2. Click Clear to clear all field values.
   - The Clear button is used to clear the field values for all the Schedule tabs:
     - When creating a new schedule.
     - While creating a schedule that is not yet added to the module or chain.
3. Complete the fields on the four Schedule sub-tabs.
   - The Schedule sub-tabs are described in the following topics.
4. To save the settings and add the schedule to the module or chain, click Add.
   - AppWorx adds the schedule to the list on the top of the Schedule tab. The Schedule will be highlighted to show that it is selected for the module or chain.
5. To create additional schedules, click Clear.
Updating and Deleting Schedules

Selecting a schedule from the list will show its detail in the Schedule tabs. You can update the schedule’s fields and click Update to update or click Delete to delete the schedule.

Running Multiple Modules/Chains on the Same Schedule

Schedules are not AppWorx objects. They are particular to a single module or chain. However, you can utilize one or more identical schedules on multiple modules and/or chains.

To run multiple modules/chains on the same schedule(s):

1. Create a chain with the desired schedule(s).
2. Add all the modules/chains you wish to run on the schedule to the chain.
3. Select the Multi thread option for each component in the chain.

Scheduling Behavior After an Outage

AppWorx has built-in logic to help you recover quickly after a system crash or an outage. By checking the value of the Next run date field in each module and chain’s schedule(s), AppWorx determines when to run the job next.

If AppWorx has been down past midnight, when the master is restarted, a master startup trigger updates the Next run date as follows:

- A module or chain that has a next run date prior to the current day, and which is scheduled for days, will be changed so that:
  - If the module or chain is eligible to run on its next run date, its next run date will not be modified.
  - If the module or chain is not eligible to run on its next run date, then the value in the Next run date field will be incremented by one day and the above check will be repeated until the next run date is today.

These checks will cause modules and chains that would normally have started during the outage time period to start immediately when the master recovers. Conversely, modules and chains that would not have started will not start until their normal start times.

- A module or chain that has a next run date prior to the current day and has multiple start times will be set to the first available of the multiple start times.

Note: These actions will only occur upon startup of the master, and will only affect modules and chains that are scheduled for a date prior to the current day. If a module or chain should have run several times during an outage, AppWorx runs it once and returns it to its current schedule.
10.2 Entering General Information for Schedules

Use the Schedules General sub-tab to name a schedule and set a start date and time and execution options to it.

Use the Schedules General sub-tab to name a schedule and set a start date and time and execution options to it.

Procedure

To set a schedule when a module or chain should run:

1. From the Schedules General sub-tab, complete the fields using the information in Table A.
2. Enter values on the other Schedule sub-tabs.
3. To save the settings and add the schedule to the module or chain, click Add for a new schedule or Update for an existing schedule.

Table A. The Fields on the Schedules General tab

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the schedule.</td>
</tr>
<tr>
<td>Frequency</td>
<td>Displays the frequency type and scheduled days selected on the Frequency sub-tab.</td>
</tr>
<tr>
<td>Start time(s)</td>
<td>The time, in 24 hour format (HH:MM), when the module/chain will be submitted to a queue. This does not indicate the exact run time—system load and conditions can affect run times. By default, AppWorx will set the time to 00:00 (midnight). You can enter as many times as you wish.</td>
</tr>
</tbody>
</table>
**Inactivating a Schedule**

If you want to inactivate a schedule for a module or chain without deleting it you can postdate the value in the **Next run date** field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next run time</td>
<td>You may enter a one-time override value here in 24 hour format (HH:MM) with multiple start times. The module/chain will run once, then revert to schedule.</td>
</tr>
<tr>
<td>Next run date</td>
<td>The next date AppWorx will evaluate the schedule to see if it is eligible to run. This field must be filled in for the module or chain to run automatically. A date in the past will cause the chain to run immediately. When the module or chain runs, AppWorx updates this field based on the information in the other scheduling fields.</td>
</tr>
<tr>
<td>Skip calendar</td>
<td>Used to schedule days not to run, such as holidays, end of month processing dates, and end of fiscal quarter processing dates. AppWorx will not run the module or chain on the dates included in the selected calendar. For information on defining calendars, see topic 10.6 Defining Calendars.</td>
</tr>
<tr>
<td>Time zone</td>
<td>The time zone the schedule will run under. This field is only active if the time zone has been set for the master. Select a standard time option if your computer does not adhere to daylight savings time. UTC refers to the Universal Time Coordinate, which is used for the synchronization of computers on the Internet. If you do not select a time zone the chain will run based on the master’s setting. For additional time zone information, see topic 17.4 Setting the Master’s Time Zone.</td>
</tr>
<tr>
<td>Queue</td>
<td>Identifies the queue to which the module or chain will be submitted. When applied to a chain, this setting overrides the queue setting for the chain’s components</td>
</tr>
<tr>
<td>Requestor</td>
<td>This field indicates the user login that the module or chain components will run under. You must have the All Chain Requestors user option assigned to you by your AppWorx administrator to select a different user in this field. If ‘No selection’ is picked, the module/chain will run under the DBA user options.</td>
</tr>
<tr>
<td>Agent</td>
<td>When running on this schedule, the module or chain will run on the agent or group assigned here. If ‘No selection’ is picked, the job will run on the agent selected on the module’s General tab.</td>
</tr>
<tr>
<td>Start date</td>
<td>The earliest date on which the module/chain is eligible to run.</td>
</tr>
<tr>
<td>End date</td>
<td>The latest date on which the module/chain is eligible to run.</td>
</tr>
<tr>
<td>Last run time</td>
<td>This read only field displays the date the module or chain last ran.</td>
</tr>
</tbody>
</table>
10.3 Entering Schedule Frequencies

Using the schedule’s Frequency sub-tab, you can select a unit of time, and options. The unit you select determines the options that are available.

Using the schedule’s Frequency sub-tab, you can select a unit of time, and options. The unit you select determines the options that are available.

Procedure

To set a schedule’s frequency:

1. From the Frequency sub-tab, select a unit of time from the Units box and its options. Unit descriptions and their details are listed in Table A.

2. Enter values on the other Schedule sub-tabs.

3. To save the settings and add the schedule to the module or chain, click Add for a new schedule or Update for an existing schedule.

Table A. The unit descriptions on the Frequency sub-tab.

<table>
<thead>
<tr>
<th>Type</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minutes</td>
<td>• Enter a number of minutes, hours, or days.</td>
</tr>
<tr>
<td>Hourly</td>
<td>• Select Weekdays, Every day, Selected days with the boxes checked for the days you wish to include.</td>
</tr>
<tr>
<td>Daily</td>
<td>• Select Weekdays, Every day, Selected days with the boxes checked for the days you wish to include.</td>
</tr>
<tr>
<td>Weekly</td>
<td>• Enter a number of weeks.</td>
</tr>
<tr>
<td></td>
<td>• Enter a date in the Reschedule from field (see below).</td>
</tr>
<tr>
<td></td>
<td>• Select Weekdays, Every day, Selected days with the boxes checked for the days you wish to include.</td>
</tr>
<tr>
<td></td>
<td>• Select an Exception method option (see below).</td>
</tr>
</tbody>
</table>
Avoiding Schedule Creep with the Reschedule From Option

After a module or chain using the Weekly or Monthly frequency runs, the values in the following fields are evaluated to determine a new Next run date on the General sub-tab.

- Every ___ Week for schedules with a Weekly frequency, or Every ___ Month for schedules with a Monthly frequency.
- The check boxes for each day of the week.
- The date in (dd:mm:yyyy) form in the Reschedule From field (displayed only when the Weekly and Monthly options are selected).

The Reschedule From field is used essential for creating scheduling that run on days such as the first Friday after the 15th of every month. If left blank, the chain is rescheduled based on the day it ran, and you can get ‘schedule creep’, where the value for the Next run date moves back or ahead several days each time the chain runs. For an example of how this option is used, see the example titled “Run a Chain the First Wednesday After the Fifteenth of Each Month at 11 P.M.” in topic 10.7 Example Schedules.
10.4 Entering Schedule Exceptions

Using the Schedules Exceptions sub-tab, you can exclude a regularly scheduled running of a module or chain.

Adding Exceptions

To add an exception to an existing schedule:

1. From the Schedule’s Exceptions sub-tab, enter the date of the exception (DD-MM-YYYY) in the Date field.
2. Enter a time (HH:MM) or select one from the drop-down list.
   - The times available for the drop-down list are based on the times entered in the Start time(s) field on the Schedule’s General tab.
3. Click on the Add button.
   - AppWorx adds the exception to the table.

You can select defined exceptions and update or delete them using the appropriate button.
10.5 Specifying Prompt Values for Schedules

You can override a prompt's default value by entering a value in the Value column on the schedule's Prompts sub-tab.

There may be times when you wish to override the default value for one or more prompts when a module or chain is called by a schedule. To do this, enter the new value in the Value column.

**Note:** Prompts cannot be added, updated, or deleted from this sub-tab.

The Prompts sub-tab displays:

- A table that displays the prompts, defined on the Prompts tab for the module or chain. It includes an override column, where the value from the Default value field can be overridden when this module (or chain) is run with the selected schedule.
- Read-only details for the prompts, as they were defined on the Prompts tab for the module or chain.

**Procedure**

To override the default value when a module is called by a schedule, enter a value in the Value column (see Figure A) and click **Apply**.

If the prompt uses a data type that includes an appropriate SQL statement in its definition, the **LOV** button will be active. Click it to open the List of Values window (see Figure B). Double-click a value to add it to the Value column.

---

**Figure A. The Schedule’s Prompts sub-tab**

**Figure B. The List of Values window.**
If the prompt uses a data type that includes an appropriate SQL statement in its definition, the MS button will be active. Click it to open the Multiple Selection Values window (see Figure C), and select one or more values.

Figure C. The Multi Selection Values window.
10.6 Defining Calendars

Calendars are used to schedule modules and chains to run or not run. A calendar specifies a list of dates such as holidays, end of month processing dates, and end of fiscal quarter processing dates.

Sometimes there are days when information is handled in special ways. In AppWorx, you can specify these days by creating calendars. You can schedule modules and chains, to run or not run on the days specified by a calendar. Calendars are useful for specifying a set of dates such as holidays, end of month processing dates, and end of fiscal quarter processing dates.

![Image of a calendar interface]

**Figure A.** To define a calendar, select the days you wish to include.

**Note:** AppWorx roles control access to calendars. If you do not have access to them, see your AppWorx administrator.
Procedure

To define a calendar:

1. From the Calendars selector window, click **New**.
   
   AppWorx opens the Calendars window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Enter a name and description for the calendar.

3. Select the dates to be included in the calendar (these could be run or skip dates, depending on the use of your calendar).
   
   Use the arrow buttons to scroll by month and the double arrow buttons to scroll by six month increments.
10.7 Example Schedules

Below are several example schedules. Use them to help you set up your own schedules.

Run a Chain on Monday, Wednesday, and Friday at 6 A.M.

With the schedule in Figure A assigned to the chain, it is eligible to execute:

- Automatically—the **Next run date** ‘04-05-2002’ is set.
- At 6 A.M. ‘06:00’ is entered in the **Start time(s)** field.
- Every Monday, Wednesday, and Friday. On the Frequency sub-tab:
  - Daily is selected in the **Units** box.
  - ‘1’ is entered in the Every ___ Days field.
  - The **Monday**, **Wednesday**, and **Friday** boxes are checked.

![Figure A](image)

*Figure A. This chain runs on Monday, Wednesday, and Friday at 6:00 A.M.*

Run a Module Every Day, Except Holidays, at 9:30 P.M.

With the schedule in Figure B assigned to the module, it is eligible to execute:

- Automatically—the **Next run date** ‘04-05-2002’ is set.
- At 9:30 P.M. ‘21:30’ is entered in the **Start time(s)** field.
- Every day except the days specified in the Holidays calendar:
  - Daily is selected as the type on the Frequency sub-tab.
  - The Holidays calendar is being used as a skip calendar, the module will not run on days in the ‘Holidays’ list (for example, 4 July, 25 December).
Run a Module on Holidays Only, at 11:59 P.M. Eastern Standard Time

With the schedule in Figure C assigned to the module, it is eligible to execute:

- Automatically—the **Next run date** ‘04-05-2002’ is set.
- At 11:59 P.M.–‘23:59’ is entered in the **Start time(s)** field.
- On Eastern Standard Time—‘EST’ has been selected in the **Time zone** field.
- Only on the days specified in the ‘Holidays’ calendar. The **Next run date** ‘04-05-2002’ is updated automatically by the ‘Holidays’ calendar.
Run a Chain the First Wednesday After the Fifteenth of Each Month at 11 P.M.

With the schedule in Figure D assigned to the chain, it is eligible to execute:

- Automatically—the **Next run date** ‘04-17-2002’ is set.
- At 11:00 P.M.—‘23:00’ is entered in the **Start time(s)** field.
- On the first Wednesday of each month. On the Frequency sub-tab:
  - **Monthly** is selected in the **Units** box.
  - ‘1’ is entered in the **Every ___ Months** field.
  - The **Wednesday** box is checked.
  - ‘04-15-2002’ is entered in the **Reschedule from** field.

Without the **Reschedule from** date, one month would be added to the date the chain ran. For example, if the chain ran on the 17th of the month, AppWorx would start looking for the first Wednesday from the 17th of the following month. This would ‘creep’ the **Next run date** for the following month ahead to the 17th day of the month, possibly skipping the first Wednesday after the 15th.

---

**Figure D.** This chain runs on the first Wednesday after the fifteenth of each month at 11 P.M.
Run a Chain Weekdays at 8 A.M., 9 A.M., 12 Noon, 5 P.M. and 11:45 P.M.

With the schedule in Figure E assigned the chain, it is eligible to execute:

- Automatically—the Next run date ‘04-05-2002’ is set.
- At 8:00 A.M., 9:00 A.M., 12:00 noon, 5:00 P.M., and 11:45 P.M—these times are entered in the Start time(s) field.
- Every weekday. On the Frequency sub-tab:
  - Daily is selected in the Units box.
  - ‘1’ is entered in the Every ___ Days field.
  - The Weekdays option is selected.

*Figure E. This chain runs weekdays at 8 A.M., 9 A.M., 12 Noon, 5 P.M. and 11:45 P.M.*
10.8 How AppWorx Determines Eligibility

Before running a scheduled module or chain, AppWorx checks its schedules, predecessor statements, and BEFORE conditions. Once a chain runs, AppWorx checks the eligibility of each chain component.

Before running a scheduled module or chain, AppWorx checks its:

- Schedules
- Predecessor statements
- BEFORE conditions.

Once a chain runs, AppWorx checks the eligibility of each chain component.

Checking Scheduling Information

AppWorx reviews the module or chain’s schedules to determine if it is eligible to run on a given day and at a given time. The chain in Figure A has one schedule. It is eligible to run each day at 12:05 A.M., Monday through Friday.

If a chain is not eligible to run based on its schedule, AppWorx terminates the evaluation process and does not check for predecessor statements.
Checking Predecessor Statements

If there are predecessor statements assigned to a module or chain, they must be evaluated as true before it can run. In Figure B, the chain is eligible to run only if the ORDERS chain has completed successfully.

![Figure B. The Predecessors tab for a chain.](image)

If a chain is not eligible to run based on its predecessors, AppWorx terminates the evaluation process and does not check for BEFORE conditions.
Checking Conditions

If all of the predecessor statements for a module or chain are met, AppWorx checks the BEFORE conditions. If no BEFORE conditions prevent it from running, AppWorx runs the module or chain. In Figure C, the chain is eligible to run unless the FTP module is in the Backlog.

If a module or chain is eligible to run, it is submitted to the Backlog as shown in Figure D. If it is not eligible to run, AppWorx does not display it in the Backlog.
Checking Component Eligibility

If a chain is eligible to run, AppWorx checks the schedule (see Figure E) of each of the chain elements for:

- The Active option
- A run or skip calendar
- The days of the week the module is eligible to run
- Predecessor statements
- BEFORE conditions

![Figure E. You can view each chain component’s schedule information on the General sub-tab.](image-url)
11

Adding Prompts to Modules and Chains

11.1 Introduction to Adding Prompts to Modules and Chains ......................................................... 248
11.2 What the Users See .................................................................................................................. 250
11.3 Adding, Updating, and Deleting Prompts .............................................................................. 252
11.4 Setting Minimum and Maximum Values for Prompts .......................................................... 254
11.5 Defining Data Types ............................................................................................................... 256
  11.5.1 Searching Your Database with SQL Statements .......................................................... 258
  11.5.2 Using Referential Checking .......................................................................................... 260
11.6 Adding a List of Values Prompt ............................................................................................. 262
11.7 Adding a Multi Select Prompt ............................................................................................... 264
11.8 How Multi Select Prompts Work .......................................................................................... 266
11.9 Copying Prompts from Another Module or Chain ................................................................. 268
11.10 Passing Values Through a Chain with Prompts ................................................................. 270
  11.10.1 Example: Passing Values Through a Chain with Prompts ....................................... 272
  11.10.2 Using Prompt Values in Chain Component Aliases ................................................... 278
11.1 Introduction to Adding Prompts to Modules and Chains

Prompts pass user input to the program run by a module. They can be defined for modules, chains or chain components. When you create a prompt, you assign it a data type. Data types define the format of the data that will be used in the prompt.

If the program called by a module requires arguments, you can use prompts to request data that will be passed to the program. For example, if a module runs a report program that accepts a start date and end date, you can create prompts that let the user enter start and end dates for the report. If the module is submitted on an ad hoc basis, the user enters values for the prompts at the time they submit the module. If the module is run as part of a chain, values for the prompts are entered at the time the module is added to the chain. You can define up to 99 prompts per module.

Types of Prompts

There are four types of prompts you can create. The type of prompt determines how the information is entered. The different types of prompts are described below.

<table>
<thead>
<tr>
<th>Prompt Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default - cannot change</td>
<td>The prompt has a default value that users are not allowed to change.</td>
</tr>
<tr>
<td>Fill-in</td>
<td>The prompt may or may not have a default value. Users are allowed to change the value.</td>
</tr>
<tr>
<td>Single selection from a list</td>
<td>Users may select one (and only one) choice from a predefined list of possible values.</td>
</tr>
<tr>
<td>Multiple selection from a list</td>
<td>Users may select one or more choices from a predefined list of possible values.</td>
</tr>
</tbody>
</table>

Prompts Defined by Data Types

When you create a prompt, you assign it a data type. The data type defines the format of the data that will be used in the prompt. There are three basic data types that are commonly used: character, number, and dates. A data type can also incorporate a SQL statement that searches the database and returns a set of values that can be selected for a prompt. For information on defining data types, see topic 11.5 Defining Data Types.

Working Smarter by Copying Prompts

If you have defined prompts for a module or chain, you can use them in another module or chain using the Copy button.
Passing Values through a Chain

You can pass prompts from a chain to one or more chain components using numeric substitution variables. The values can be used in prompts, condition, and aliases. The user can enter values in one place and have them automatically used in one, several, or all of the modules in a chain. This technique is described in topic 11.10 Passing Values Through a Chain with Prompts.
11.2 What the Users See

To respond to a prompt in the module(s) you create, a user can accept the default value if one is displayed, enter a value, or click the LOV (List of Values) or MS (Multi Select) button to select a value from a list. Below are some examples of the screens they will use.

If you are creating a module that users will run on an ad hoc basis, the prompts you create will allow users to enter values, thereby customizing the module. The module then passes these values to the program. A prompt may accept one or multiple values. For more information on submitting modules, see topic 3.2 Requesting Modules and Chains.

**Responding to an LOV Prompt**

If the user selects an LOV (List of Values) prompt, they can click the LOV button to display a list of values as shown in Figure A. To add an LOV prompt, see topic 11.6 Adding a List of Values Prompt.

![Figure A. Users can select a value by selecting the LOV button.](image-url)
Responding to an MS Prompt

If a user selects an MS (Multi Select) prompt, they can click the MS button to display a prompt that accepts multiple values as shown in Figure B. To add an MS prompt, see topic 11.7 Adding a Multi Select Prompt.

Figure B. Users can select multiple values by selecting the MS button.
11.3 Adding, Updating, and Deleting Prompts

To define a prompt for a module, chain or chain component, select the Prompts tab, fill in the fields on the bottom of the tab, and click Add. To update or delete a prompt, select the prompt and edit the fields or select Delete. To reorder prompts in a module or chain, use the arrow buttons to the right of the list of prompts.

You can add one or more prompts to a module, chain or chain component.

**Figure A.** You define a prompt for a module on the module’s Prompts tab.

### Adding Prompts to a Module or Chain

To add a prompt to a module or chain:

1. Select the Prompts tab for a module, chain or chain component as shown in Figure A.
2. Complete the fields using the information in Table A.
3. To save the settings and add the prompt to the module or chain, click Add. AppWorx adds the prompt to the list on the top of the Prompts tab.
4. If you wish to reorder the prompts in a module or chain, use the arrow buttons to the right of the list of prompts.

### Updating and Deleting Prompts

To update or delete a prompt, select the prompt and edit its fields or select delete.
Table A. Field and option descriptions for Prompts tab

<table>
<thead>
<tr>
<th>Field/Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Defines the type of data validations that will be used. For information on defining data types, see topic 11.5 Defining Data Types.</td>
</tr>
</tbody>
</table>
| List of values Multi select| Some data types include SQL statements that allow you to select prompt values from a list. The following options are active when the selected data type includes a SQL statement:  
  • List of values: allows you to select a single value.  
  • Multi select: allows you to select one or more values. Developers use these to assign values to the Default value field using the Select button to the right of that field. Operators use these to assign values to prompts when requesting the module/chain. |
| Description                | Text displayed in the Submit window when the module/chain is requested and on the Components tab if the module (or chain) is added to a chain. |
| Variable name              | The variable name for the prompt. For example, if you want to ensure that the prompt is assigned to the variable ‘time’ (used in a SQL script, say), put ‘time’ in the Variable Name field. Variable names are used differently by different program types (up to 100 characters). |
| Default value              | The default value is displayed in the Default column. If Allow changes is checked, the user can enter their own value. You can use substitution variables in this field. |
| Minimum Maximum            | Sets the minimum and maximum value a user can enter in response to the prompt. You can type in values, use the drop-down box to select a substitution variable, or select from a list of values if the Select button is active. The Select button is active only when the data type selected includes a SQL statement. If a Min value and Max value have been defined for the data type, they will be displayed in these fields. For more information on these fields, see topic 11.4 Setting Minimum and Maximum Values for Prompts. |
| Max length                 | Sets the maximum number of characters a user can enter in response to the prompt (up to 512).                                                  |
| Value required              | When selected, this prompt must have either a default or user entered value when the module/chain is executed.                              |
| Allow changes              | When selected, the user can enter a value for the prompt. If you do not select this option, the user cannot change the default value.         |
| Upper case                 | When selected, text entered for the prompt will be forced to upper case at the time the module is submitted. The default setting is determined by the prompt’s data type. |
| Quote char                 | The quote character is required for some program types if null values or space characters are allowed in the prompt. The quote character is placed before and after the value entered for the prompt. Single and double quote characters are common choices. |
11.4 Setting Minimum and Maximum Values for Prompts

If while defining a prompt, you select a data type with minimum and maximum values defined, those values automatically populate the prompt’s Minimum and Maximum fields. The values serve as guidelines and can be changed.

The Minimum and Maximum fields specify a range that users can enter in response to a prompt. If while defining a prompt, you select a data type with a Min value and Max value defined, those values automatically populate the prompt’s Minimum and Maximum fields (see Figure A).

The values serve as guidelines and can be changed. You can edit them by:

- Typing in new values.
- Selecting a substitution variable using the drop-down list or icon.
- Select from a list of values if the Select button is active. The Select button is active only when the data type selected includes a SQL statement.
11.5 Defining Data Types

Data types are used to define the format that will be used in the prompts for modules, chains and chain components.

If the program run by a module accepts parameters, you can create a prompt for each parameter in the program. When you create prompts, you define its characteristics. From the Prompts tab for modules, chains, and chain components, you select (and/or define) the data type. You create data types for the express purpose of defining module prompts.

AppWorx ships with a number of predefined data types. However, you will probably want to create your own data types to meet the needs of your operation.

**Note:** AppWorx roles control access to data types. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To define a data type:

1. From the Data Types selector window, click **New**.
   
   AppWorx opens the Data Types window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.
   
2. Complete the fields using the information in Table A.
Table A. Field descriptions for the Data Types window.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A 100-character description for the data type.</td>
</tr>
<tr>
<td>Type</td>
<td>General type–can be character, date, or number. For example, a “Divisions” data type would use character; while a “Product Count” data type would use number.</td>
</tr>
<tr>
<td>Min value</td>
<td>These fields define the Minimum and Maximum for the prompts. For more information on these fields, see topic 11.4 Setting Minimum and Maximum Values for Prompts.</td>
</tr>
<tr>
<td>Max value</td>
<td></td>
</tr>
<tr>
<td>Max length</td>
<td>Defines the maximum length of the data in characters. The value may not exceed 512 characters.</td>
</tr>
<tr>
<td>Date input format</td>
<td>These two drop-down list boxes are displayed only when the data type is Date. They define the input format for a date entered in a prompt and the output format at the time the program is executed. They are used for date format verification and conversion.</td>
</tr>
<tr>
<td>Date output format</td>
<td></td>
</tr>
<tr>
<td>Referential checking</td>
<td>When this option is selected, AppWorx can reference values entered for previous prompts in the same module. For more information on using referential checking, see topic 11.5.2 Using Referential Checking.</td>
</tr>
<tr>
<td>Upper case</td>
<td>When this option is selected, AppWorx converts all values to upper case characters.</td>
</tr>
<tr>
<td>SQL statement</td>
<td>Contains a select statement which provides a list of values for a prompt. The SQL statement can be up to 2000 characters long. It must return two values (in columns). Each value can be no more than 100 characters long. The first value displays the prompt’s value. The second value provides additional information about the first value. Although SQL statements can return values of various types, including numbers and dates–AppWorx evaluates them all as characters. Therefore, you must return your values as text. To convert a number to text, use the to_char(so_pt_value) function. In Figure A, the to_char(deptno) function converts the department number to text.</td>
</tr>
<tr>
<td>Check SQL</td>
<td>Executes the SQL statement and displays the output in the SQL View window. For more information see topic 11.5.1 Searching Your Database with SQL Statements.</td>
</tr>
</tbody>
</table>

Subtopics

The subtopics that follow describe how to pull values from your corporate database.

11.5.1 Searching Your Database with SQL Statements

11.5.2 Using Referential Checking
11.5 Defining Data Types

11.5.1 Searching Your Database with SQL Statements

You can use SQL statements in your data types to pull values from your corporate database.

You can use a SQL statement to pull values from your corporate or non-AppWorx database. The values can be used to provide a list of values that can be entered for a prompt.

![SQL View](image)

**Figure A.** The modules data type. When you check a SQL statement, the results are displayed in the SQL View window.

The Dept_Name data type shown in Figure A uses a database grant to display a list of departments in the `my_db` database. You could also use a database link to access the `my_db` database as shown below.

```sql
SELECT dname, to_char(deptno) FROM DEPT@my_db
```

For information on grants and database links, see your Oracle documentation or database administrator.

Checking SQL Statements

If you add a SQL statement to a data type, you can test the results of the statement by clicking the **Check SQL** button. AppWorx displays the results in the SQL View window shown in Figure A.
11.5 Defining Data Types

11.5.2 Using Referential Checking

Referential checking lets you use the value(s) specified for one or more previous prompts to limit the choices in a subsequent prompt. Referential checking provides a much more meaningful List of Values prompt for operators.

One of the powerful features of AppWorx is the ability to define List of Values (or LOV) prompts for modules and chains. When an end user requests a module, an LOV prompt will provide a list of appropriate values from which to choose. All an operator needs to do is pick a value from the list. Referential checking makes a List of Values all the more useful by providing operators with a list that takes into consideration values entered in prior prompts.

Setting Up Referential Checking

To set up referential checking, you must define a data type with:

- A SQL statement that defines a List of Values for a prompt.
- A reference in the SQL statement to another prompt using either:
  - A numeric substitution variable defined by a prompt by number. For example {#1}. For more information in numeric substitution variables, see topic 11.10 Passing Values Through a Chain with Prompts.
  - A variable name defined in the Variable name field for the first prompt. For example {#dept_name}.
  - The Referential checking option checked.

Example: Selecting a Job Title by Department

You may have a module named REPORT that requires two prompts—Department and Job Title. You can set up the first prompt as a List of Values to provide a list of valid departments from which an operator can choose. Using referential checking, you can set up the second prompt as a List of Values to provide a list of job titles from the department chosen in the first prompt.

Setting up REPORT in this way increases its reliability by preventing an operator from submitting the module with a mismatched department and job title. At the same time, it makes it easier for them to submit the module, because they pick from a list rather than having to remember or guess departments and job titles.

The Department data type (used for prompt 1) requires a SQL statement to return a list of Departments (there is no referential checking here):

```
SELECT DISTINCT dname to_char (deptno) from DEPT
```
The Job Title data type requires a SQL statement to return a list of job titles relative to the department chosen in the first prompt:

```
SELECT DISTINCT EMP.job, EMP.job FROM EMP, DEPT
WHERE DEPT.dname = '{#1}'
AND EMP.deptno = DEPT.deptno
```

Notice the reference to prompt #1 (in bold). When the Job Title data type is used in the second prompt (and the data type's Referential checking box is checked) it will reference REPORT's first prompt. The `{#1}` will be replaced with the value from REPORT's first prompt and the SQL statement will return a list of job titles for the chosen department.

But, what if REPORT is modified later to have additional prompts and Department isn't the first prompt anymore? The reference `{#1}` will still refer to the first prompt and the Job Title prompt will no longer work correctly. We can solve this problem by referring to the Department prompt by name rather than by number. We can add a name for the Department prompt in the prompt's Variable name field, for example `dept_name`. In the data type SQL statement for Job Title, we can use the prompt's name in the reference rather than its number:

```
SELECT DISTINCT emp.job_title, dept.dept FROM emp, dept
WHERE dept.dept = '{#dept_name}'
AND dept.deptno = emp.deptno
```

**Cannot Check SQL**

If you use referential checking, you will not be able to check the data type's SQL statement with the Check SQL button because the SQL statement contains references that cannot be resolved (the prompt values referenced by `{#number}` or `{#variable_name}` don't exist yet). If you click Check SQL, a window appears to remind you why the statement cannot be checked.
11.6 Adding a List of Values Prompt

To add a List of values prompt, select a data type that includes a SQL statement, which generates a list.

List of values (LOV) prompts allow users to select a single value from a list of values. The list is generated by a SQL statement defined for the data type selected for the prompt. For information on creating data types, see topic 11.5 Defining Data Types.

Figure A. The Multiple Select/List of Values options include a data type with an appropriate SQL statement.

The value the user selects from the list is stored in a table, then passed to the program run by the module at the time the module is executed. LOV prompts are useful for letting a user select values from a predefined list such as departments, part numbers, sales regions, states, or countries. If a user types in a value for an LOV prompt, the value is checked against the list generated by the SQL statement.
Procedure

To add a list of values prompt:

1. From the Prompts tab for a module or chain as shown in Figure A, select a data type that includes the appropriate SQL statement in its definition. Figure B shows the SQL statement for the Modules data type. AppWorx activates the List of values and Multi select options when a data type with a SQL statement is selected.

2. Select the List of values option. If you wish to create a multiple selection prompt, select Multi select. Multiple selection prompts are described in topic 11.7 Adding a Multi Select Prompt.

3. To add a default value for the prompt, click the Select button to the right of the Default value field. AppWorx displays the List of Values window shown in Figure A.

4. Highlight the values you want to use and click OK, or simple double-click the value. AppWorx displays the value in the Default value field.

5. Complete the other fields as explained in topic 11.3 Adding, Updating, and Deleting Prompts.

6. To save the settings and add the prompt to the module or chain, click Add.
11.7 Adding a Multi Select Prompt

To add a multi select prompt, select a data type that includes a SQL statement.

Multi select (MS) prompts allow users to select one or more values from a list of values. The list is generated by a SQL statement defined for the data type and selected for the prompt. For information on creating data types, see topic 11.5 Defining Data Types.

![Figure A. A data type with a SQL statement activates the Multiple Select option for a prompt.](image)

The values the user selects are stored in a table, then a reference number is passed to the program run by the module at the time the module is executed. Multiple selection prompts are useful for letting a user select a set of values for a prompt.

**Procedure**

To add a multi select prompt:

1. From the Prompts tab for a module or chain as shown in Figure A, select a data type that includes the appropriate SQL statement in its definition. Figure B shows the SQL statement for the Modules data type.

   AppWorx activates the **List of values** and **Multi select** options when a data type with a SQL statement is selected.

2. Select the **Multi select** option.

3. Enter a description.
4. To add one or more default values for the prompt, click the **Select** button to the right of the **Default value** field.

AppWorx displays the Multiple Selection Values window shown in Figure B. The values displayed in the **Assigned** column will be the default values for the prompt. The user will be able to add and delete values between the columns.

**Note:** First and Next buttons are displayed when there are more than 100 values in the **Unassigned** column. Clicking Next imports the next 100 values (101-200) into the **Unassigned** column.

5. Select the values you want to use.

For information on selecting values, see topic 2.2 *Working in the AppWorx Windows.*

After selecting default values, AppWorx displays a reference number in the **Default value** field.

6. Complete the other fields as explained in topic 11.3 *Adding, Updating, and Deleting Prompts.*

7. To save the settings and add the prompt to the module or chain, click **Add.**
11.8 How Multi Select Prompts Work

When you submit a module with a multi select prompt, AppWorx writes a record in the `so_multiselect` database table using a unique reference number for each selected value. The reference number is passed to the script or application like any other prompt value. In order for a script or application to use the values, it must retrieve the records from the database.

When you submit a module with a multi select prompt, AppWorx writes a record in the `so_multiselect` database table using a unique reference number for each selected value. The reference number is passed to the script or application like any other prompt value. In order for a script or application to use the values, it must retrieve the records from the database. You can retrieve records using a run-time extension script, a program type script, or through the script or application called by the module.

![Figure A. Click MS to select one or more values.](image)

**Example**

When the `MODULE_RUN_REPORT` module shown in Figure A is requested, an end user can select one or more modules to see whether they have run within the last 24 hours. Once values are selected, AppWorx assigns a reference number to them. The reference number and values are stored in the `so_multiselect` table. The reference number is also passed to the module's script.
The **so_multiselect** table fields are described below:

<table>
<thead>
<tr>
<th>Table fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>so_ms_ref_no</td>
<td>The ID number AppWorx assigns to the multi select prompt. This number will be the same for each record generated by the prompt. It is displayed in the Default column on the Submit window. In Figure A, the ID number is 103213.</td>
</tr>
<tr>
<td>so_ms_source</td>
<td>This field simply stores an R to show that the module was requested.</td>
</tr>
<tr>
<td>so_ms_value</td>
<td>A value selected for the prompt. In Figure A, SYSTEM and TEST MODULE are the selected values.</td>
</tr>
<tr>
<td>so_ms_descr</td>
<td>The description for each selected value.</td>
</tr>
</tbody>
</table>

When the module executes, it runs the **jobsrun.sql** script shown below. The **jobsrun.sql** script accesses the **so_multiselect** table and returns the reference number and its values.

```sql
set verify off
title 'Modules run prior day'
spool &so_outfile
select so_module Module, so_status_name Status, to_char(so_job_finished, 'mm/dd/yyyy hh24:mi:ss') Finished
from so_multiselect b, so_job_history a
where so_job_finished between trunc(sysdate-1) and trunc(sysdate)
and a.so_module = b.so_ms_value
and b.so_ms_ref_no = &ms_list
/
spool off
```

When the module completes, you can view its output with the File Viewer (see Figure B).

---

**Figure B.** The output from the MODULE_RUN_REPORT module.
11.9 Copying Prompts from Another Module or Chain

To copy prompts from a module or chain, click the Copy button.

If you have defined prompts for a module and want to use them for another module, you can click the Copy button and select a module or chain to copy from. When you copy prompts, they are added to the prompts already defined for the module.

*Figure A.* You can copy prompts between modules with the Copy button.
Procedure

To copy prompts from one module or chain to another:

1. From the Prompts tab for a module, chain, or chain component click the Copy button.
   AppWorx displays the Copy prompts window shown in Figure A.

2. Select the module or chain containing the prompts you want to copy.
   You can filter the list by selecting an application. Applications specify a group where modules are assigned. Only the applications modules and chains assigned to you via roles will be displayed.
   You can type the first few letters of a module or chain’s name in the Search field, and AppWorx will highlight it. If two modules or chains start with the letter you type, AppWorx highlights the first one.

3. Click OK.
   AppWorx adds the prompts to the module/chain.
11.10  Passing Values Through a Chain with Prompts

To pass values through a chain, add prompts to the chain and enter numeric substitution variables in its modules.

You can pass values from a chain to its modules. The values passed to a chain come from fill-in prompts added to the chain. You can define up to 99 values. The values can be used in prompts, conditions, and aliases. By using prompts, the user can enter values in one place and have them automatically used in one, several, or all of the modules in a chain.

The 'numeric substitution variables' #1, #2, #3......#99 are the link between the chain and its modules. As you add prompts to the chain, AppWorx numbers them sequentially. When you add modules to the chain, you use the numbers assigned to the prompts in the chain in the module prompt fields and conditions.

Figure A shows an example of a chain called ACCOUNTING_CHAIN. The five prompts in the chain allow the user to enter an accounting period, the first and last days of the accounting period, and a region and department. AppWorx stores these values in substitution variables #1, #2, #3, #4, and #5.

These values are passed to the modules REPORT_A, REPORT_B, and REPORT_C in the ACCOUNTING_CHAIN by entering the appropriate substitution variable. In the module REPORT_A, the value for the last day of the period is also used in a condition. The statement checks to make sure the current day is after the last day of the period. If it is not, the report is aborted. REPORT_A also uses the date substitution variable from the 'First day of period' prompt as its alias.

Prompt values can be used in any combination in any number of modules. For example, REPORT_C uses only prompts #4 and #5. You also can mix numbered prompts with other types of prompts.

Note: You can also use this method to pass values from a chain to additional chains nested within it.

Subtopics

The subtopics that follow provide an example of how values can be passed from a schedule module to the modules in a chain and to a module’s alias:

11.10.1 Example: Passing Values Through a Chain with Prompts

11.10.2 Using Prompt Values in Chain Component Aliases
Figure A. You can pass values to modules in a chain using numbered substitution variables.
11.10 Passing Values Through a Chain with Prompts

11.10.1 Example: Passing Values Through a Chain with Prompts

The following example shows how prompt values can be passed through a chain.

Assume you want to create three modules that run reports against the AppWorx job history table. The reports and their matching program names are shown in the table below.

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
<th>Program Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORT_BATCH</td>
<td>Lists all jobs by queue</td>
<td>report_batch.sql</td>
</tr>
<tr>
<td>REPORT_MODULE</td>
<td>Lists all jobs by module</td>
<td>report_module.sql</td>
</tr>
<tr>
<td>REPORT_REQUESTOR</td>
<td>Lists all jobs by requestor</td>
<td>report_requestor.sql</td>
</tr>
</tbody>
</table>

Each report has been written to accept start date and end date values. When you create the modules, you will define prompts for the start and end dates. You will use substitution variables to provide default values for the dates. When you create the chain, you will create matching prompts for the start and end dates. When a user runs the chain on an ad hoc basis, they will enter the start and end dates, and AppWorx will pass the dates down to each report module in the chain.

**Step 1: Define the Data Types**

For this example, you would use the Dates data type, which comes predefined with AppWorx. Therefore, you would move on to Step 2.

You can create new data types to suit your needs. For more information, see topic 11.5 *Defining Data Types.*
Step 2: Create the Modules and Define the Prompts

To create the report modules and define the prompts:

1. Create the REPORT_BATCH module as shown in Figure A.

![Figure A. Creating the report modules](image)

2. On the Prompts tab, create two prompts, one for the start date and one for the end date, using the values shown below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Start Date Prompt</th>
<th>End Date Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Dates</td>
<td>Dates</td>
</tr>
<tr>
<td>Description</td>
<td>Enter start date:</td>
<td>Enter end date:</td>
</tr>
<tr>
<td>Variable name</td>
<td>job_start_date</td>
<td>job_end_date</td>
</tr>
<tr>
<td>Default value</td>
<td>#yesterday</td>
<td>#today</td>
</tr>
</tbody>
</table>

3. Submit the module to make sure the report runs properly.
4. Repeat steps 1 through 4 to create the other two reports.

   **Note:** Use the **Copy** button to copy the prompts from the REPORT_BATCH module for the second and third modules (see Figure B).

![Figure B. Use the Copy button to copy prompts from one module to another.](image)

**Step 3: Create the Chain and Add the Modules**

To create the chain that runs the three reports:

1. Create a chain and name it REPORTS.
2. Add the three report modules to the chain.
   
   Select the **Multi thread** option for each component so the jobs will process faster. Also be sure that Block on abort is not selected in the **Link** field. This way if one of the modules fails, it will not prevent the other jobs from running.

You can test the chain by submitting it.
Step 4: Enter the Substitution Variables for the Module Prompts

To modify the modules in the REPORTS chain to accept start and end dates passed through the chain:

1. Enter the following override values on Prompts sub-tab, for each of the chain component (REPORT_BATCH, etc.),

<table>
<thead>
<tr>
<th>Prompt</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter start date:</td>
<td>#1</td>
</tr>
<tr>
<td>Enter end date:</td>
<td>#2</td>
</tr>
</tbody>
</table>

The #1 and #2 tell AppWorx to use the values from the first and second prompts of the chain.

Figure C. Enter the #1 and #2 numeric substitution variables for each chain component.
Chapter 11: Adding Prompts to Modules and Chains

Step 5: Adding Prompts to the Chain

To add prompts to be passed through the chain:

1. From the chain’s Prompts tab, create a prompt for the start date and a prompt for the end date using the values shown below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Start Date Prompt</th>
<th>End Date Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data type</td>
<td>Dates</td>
<td>Dates</td>
</tr>
<tr>
<td>Description</td>
<td>Enter start date:</td>
<td>Enter end date:</td>
</tr>
<tr>
<td>Variable name</td>
<td>(leave blank)</td>
<td>(leave blank)</td>
</tr>
<tr>
<td>Default value</td>
<td>#yesterday</td>
<td>#today</td>
</tr>
</tbody>
</table>

A variable name is not needed because the prompt values are passed to an AppWorx module, not a program.

2. Select the Value required and Allow changes boxes.

Testing the Numbered Substitution Variables

To test the numbered substitution variables:

1. Using the Requests window, call up the REPORTS chain you just created.
2. Accept the default values for the prompts, or enter different dates.
3. Submit the chain.
4. Monitor the status of the modules from the Jobs/Explorer window.
5. When the job completes, view the output from the three reports.
Chapter 11: Adding Prompts to Modules and Chains

11.10 Passing Values Through a Chain with Prompts

11.10.2 Using Prompt Values in Chain Component Aliases

You can use a numeric substitution variable to display the value from a chain’s prompt as a component’s alias in the Backlog and History at run time.

You can use a numeric substitution variable to display the value from a chain’s prompt as one of its components’ alias. The alias is displayed in the Backlog and History at run time. To do this, add a substitution variable (inside of curly braces, see Figure A) in the Alias field for a chain component on the General sub-tab. You can combine the substitution variable with text.

**Prerequisites**

- The module/chain must be part of a chain.
- The chain must include one or more prompts.

![Figure A](image-url)  
*Figure A. You can enter a numbered substitution variable in the Alias field to display a variable alias for the module at run time.*
**Procedure**

To use the data a user enters from a prompt as an alias for a chain component:

1. Create the data types, modules, and prompts for your chain. For an example, see topic 11.10.1 Example: Passing Values Through a Chain with Prompts.

2. Enter the number in the **Alias** field on the General sub-tab of the chain component for the prompt that you would like to have displayed as the module's name in the Backlog and History.

   You must put curly braces around the prompt number to indicate that the alias includes a numeric substitution variable. You can use the numeric substitution variables by:
   - Just including the variable.
     
     For example: {#3}
   - Combining the substitution variable with text.
     
     For example: {#3}text or {#1}text{#2}
12

Assigning Predecessor Statements

12.1 Introduction to Assigning Predecessor Statements .............................................. 282
12.2 Setting Virtual Workday Start Times ................................................................. 284
12.3 Adding Predecessor Statements to Modules and Chains ................................. 286
12.4 Grouping Predecessor Statements ........................................................................ 288
12.5 Deciding Between Predecessors and CHECK HISTORY Conditions ............... 290
12.6 Example: Running a Series of Modules in a Virtual Day ...................................... 292
12.7 Example: Using Branching Logic to Determine Which Module Runs ............... 294
12.8 Example: Running on the Successful Completion of a Chain Component .......... 296
12.1 Introduction to Assigning Predecessor Statements

Predecessor statements can be added to modules, chains, and chain components. With them you can run the module or chain if a particular module or chain has succeeded, failed, or completed since the last “virtual workday” start time.

Predecessors allow you to run a job if a particular module or chain has succeeded, failed, or completed since the last virtual workday start time. They can be added to modules, chains, and chain components using the Predecessors tab shown in Figure A. The predecessor statements must be met before the module or chain will be eligible to run.

You can add, edit, or delete predecessor statements for non-running jobs in the Backlog. When you do this, the changes are particular only to that running of the job. Predecessor statements are evaluated prior to any BEFORE conditions the module or chain might have.

Setting Virtual Workday Start Times

The virtual workday is used to establish reset times for predecessor statements. You set the virtual workday start time by running the SET_VIRTUAL_DAY module. Depending on your organization’s needs, you can schedule this module to run every day, every few days, or several times each day. For more information, see topic 12.2 Setting Virtual Workday Start Times.
12.2 Setting Virtual Workday Start Times

The virtual workday is used to establish reset times for predecessor statements. You set the virtual workday start time by running the SET_VIRTUAL_DAY module. Depending on your organization’s needs, you can schedule this module to run every day, every few days, or several times each day.

The virtual workday is used to establish reset times for predecessor statements. For example, you might run a production schedule every two hours, and reset the virtual day every two hours. The current virtual workday start time is displayed to the left of the status bar on the Explorer window (see Figure A).

**Figure A.** The current virtual workday time is displayed to the right of the Status Bar on the Explorer window.

**Procedure**

You set the virtual workday start time by running the SET_VIRTUAL_DAY module. Depending on your organization’s needs, you can schedule this module to run every day, every few days, or several times each day.

**Note:** You cannot reset the virtual workday using the SET_VIRTUAL_DAY module in a chain where predecessor statements are evaluated. The SET_VIRTUAL_DAY module must finish executing before the chain is submitted.

If you want to set the virtual workday on an ad hoc basis, you can submit the module as shown in Figure B.
The SET_VIRTUAL_DAY module includes one editable prompt. The default value for the prompt is set using the #aw_now substitution variable. It returns a series of numbers that represent the current date and time.

In Figure B the default value is 20020408140809, this translates to:

- Year: 2002
- Month: April (04)
- Day of month: 8th (08)
- Time: 2:08:09 P.M. (140809)
12.3 Adding Predecessor Statements to Modules and Chains

From the Predecessors tab, you can add, edit or delete predecessor statements for modules, chains, and chain components. You can use the (+) and (-) buttons to group the predecessor statements for the module or chain.

Procedure

To add a predecessor statement to a module, chain, or chain component:

1. Select the Predecessors tab for a module, chain, or chain component. A module's Predecessors tab is shown in Figure A.

   **Note:** Predecessor statements can be added, edited, and deleted for a particular running of a module/chain before it runs in the Backlog. For more information, see topic 5.9.2 Viewing Job Predecessors.

2. Select the AND or OR option.

   • To require both this statement and the previous predecessor statement to be true for this task to run, select **AND**.
   • To require either this statement or the previous predecessor statement to be true for this task to run, select **OR**.

For the first predecessor statement, disregard the AND and OR radio buttons.
3. Select the module or chain that must have succeeded, failed, or completed since the last virtual workday start time.

You can select a module or chain from the Modules/Chains sub-tab, or a chain component from the Chain Components sub-tab.

To select a chain component from the Chain Component sub-tab, you must first select its chain (PRODSCH is selected in Figure B)

If you may be changing the components alias, it is recommended that you check the Propagate Alias box for the chain component in the chain’s definition. For more information on the Propagate Alias box, see topic 9.4 Updating Component Definitions.

4. Select an option from the statement’s drop-down box.

<table>
<thead>
<tr>
<th>Options</th>
<th>Run this task if the selected module/chain ran:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Successfully.</td>
</tr>
<tr>
<td>Success (Skip on Failure)</td>
<td>Successfully. Skip this task if the module/chain failed with a status such as DIED, ABORTED, or TIMEDOUT.</td>
</tr>
<tr>
<td>Failure</td>
<td>And failed with a status such as DIED, ABORTED, or TIMEDOUT.</td>
</tr>
<tr>
<td>Failure (Skip on Success)</td>
<td>And ran and failed with a status such as DIED, ABORTED, or TIMEDOUT. Skip this task if the module/chain ran successfully.</td>
</tr>
<tr>
<td>Complete</td>
<td>And completed, failed, or was deleted or skipped.</td>
</tr>
</tbody>
</table>

5. Click Add.

AppWorx adds the predecessor statement to the box above. When the job is run, all predecessors must be met before it can execute. Jobs that are waiting for predecessors to be met display PRED WAIT as their job status.

**Editing and Deleting Predecessor Statements**

You can highlight text in the main box and:

- Edit the values in predecessor statements’ fields to update them.
- Click the Delete button to delete one or more predecessor statements.
- Use the (+) and (-) buttons to group one or more predecessor statements.

For more information on grouping predecessor statements, see topic 12.4 Grouping Predecessor Statements.
12.4 Grouping Predecessor Statements

You can group predecessor statements using the (+) and (-) buttons. To activate the (-) button, you must highlight one or more statements, as well as their open and close parentheses.

You can group and ungroup predecessor statements using the (+) and (-) buttons. To activate the (-) button, you must highlight one or more statements, as well as their open and close parentheses. Three examples are shown below.

**Example 1**

A job with the following predecessor statements will run when either MODULE_A or MODULE_B have finished successfully and either MODULE_C or MODULE_D have finished successfully.

\[
\begin{align*}
\text{\{& MODULE\_A = \text{Success} \\
\text{OR} \\
\text{MODULE\_B = Success} \\
\text{\}} \\
\text{AND} \\
\text{\{& MODULE\_C = \text{Success} \\
\text{OR} \\
\text{MODULE\_D = Success} \\
\text{\}}}
\end{align*}
\]

**Example 2**

A job with the following predecessor statements will run when MODULE_A has finished successfully and either MODULE_B or MODULE_C has failed.

\[
\begin{align*}
\text{MODULE\_A = \text{Success} } \\
\text{AND} \\
\text{\{& MODULE\_B = \text{Failure} \\
\text{OR} \\
\text{MODULE\_C = \text{Failure} \\
\text{\}}}
\end{align*}
\]
Example 3

A job with the following predecessor statements will run when MODULE_A or MODULE_B has finished successfully. If neither MODULE_A or MODULE_B has finished successfully, but MODULE_B has failed, the job will be skipped.

MODULE_A = Success
OR
MODULE_B = Success (Skip on Failure)
12.5 Deciding Between Predecessors and CHECK HISTORY Conditions

Predecessors and CHECK HISTORY conditions are very similar. However, each offers unique advantages. It is important to understand them both and use the one that is right for your organization.

Conditions control job execution in various ways. One particular type of condition, a CHECK HISTORY condition, can be used similarly to a predecessor. For more information on conditions, see chapter 13: Working with Conditions.

Both predecessors and conditions can be added to modules, chains, or chain components. They are both used to control the execution of the module/chain they are assigned to. When modules/chains are submitted to the Backlog, AppWorx evaluates whether any predecessors or conditions cause them not to start.

With predecessors, you can run the module or chain if a particular module or chain has succeeded, failed, or completed since the last “virtual workday” start time. If the module/chain has not run, the job will wait in the Backlog with a job status of PRED WAIT. A predecessor is shown in Figure A.

![Figure A](image)

*Figure A. This module will not run until the FTP module has completed successfully this virtual work day.*

With a CHECK HISTORY condition, you can check the History to see if a module or chain has succeeded or failed within a set time period. If the module/chain has not run, the job will wait in the Backlog with a job status CONDITN WAIT. A CHECK HISTORY condition is shown in Figure B.
Although the two are very similar, each option offers its own unique advantages.

**Advantages of Predecessors**

Predecessors have several distinct characteristics. They:

- Are managed from memory. They use fewer system resources than conditions which run a SQL statement every time they are evaluated.
- Check for the module/chain since the last “virtual workday” start time.
  
  For an example of a predecessor that checks for a module since the start of the last virtual workday, see topic **12.6 Example: Running a Series of Modules in a Virtual Day**.
- Can be evaluated on two statuses that CHECK HISTORY conditions cannot.
  - Success (Skip on Failure)
  - Failure (Skip on Success)

**Advantages of CHECK HISTORY Conditions**

CHECK HISTORY conditions have several distinct characteristics. They:

- Allow you to specify a particular number of days, hours, or minutes to check for.
  
  For an example of a condition that checks for a module in a set time period, see topic **13.5.4 Checking for a Finished Job in the History**.
- Can be assigned to different timings (before a job runs, during its execution, after it has run, or when it is deleted).
- Can be assigned different actions (cancel a chain, request a module, set a substitution variable, etc.).

![Figure B. This module will not run until the FTP module has completed successfully in the last day.](image)
12.6 Example: Running a Series of Modules in a Virtual Day

In this example, predecessor statements allow you to run three modules based on the successful completion of another module that virtual day.

There may be times when you want to run a series of programs, based on the successful completion of other programs. For example, if a program “A” completes successfully that virtual day, you want programs “B”, “C”, and “D” to run. If program “A” fails, you want to skip programs “B”, “C”, and “D”. Assume that the SETVIRTUAL_DAY module is scheduled to run at different times on different days throughout the week. The logic is illustrated in Figure A.

For this example, you could create four modules:

• MODULE_A, which runs program “A”.
• MODULE_B, which runs program “B”.
• MODULE_C, which runs program “C”.
• MODULE_D, which runs program “D”.

For MODULE_B, MODULE_C, and MODULE_D, you would add the following predecessor statement:

MODULE_A = Success (Skip on Failure)

Each of the modules will run if MODULE_A completes successfully and be skipped if MODULE_A fails.

Note: You couldn’t use a CHECK HISTORY condition in this scenario because you do not know when the virtual workday starts. Therefore, you would not know how much time to check the History for.

You could easily add more complex predecessors to the modules (for example, you could only allow the module to run if one of two modules complete and another one fails).

You may choose to put a series of modules into a chain, and multi-thread them. This wouldn’t change their functionality, but it would allow you categorize them on the chain’s Components tab, and in the Jobs/Explorer window.
12.7 Example: Using Branching Logic to Determine Which Module Runs

In this example, predecessor statements allow you to determine which of two modules will run based on the successful or unsuccessful completion of another module that virtual day.

There may be times when you want the successful or unsuccessful completion of a program to determine which of two other programs will run each virtual day. For example, if a program “A” completes successfully, you want only program “B” to run. However, if program “A” fails, you want only program “C” to run. The logic is illustrated in Figure A.

For this example, you could create three modules:

- MODULE_A, which runs program “A”.
- MODULE_B, which runs program “B”.
- MODULE_C, which runs program “C”.

You would schedule each of the modules to run daily, and assign the following predecessor statements:

For MODULE_B you would add the following predecessor statement:

```
MODULE_A = Success (Skip on Failure)
```

For MODULE_C you would add the following predecessor statement:

```
MODULE_A = Failure (Skip on Success)
```

MODULE_B will run only if MODULE_A completes successfully. And MODULE_C will run only if MODULE_A does not complete successfully.

Other Methods

You could alternately use branching logic by:

- Adding two conditions to MODULE_A. If its status is equal to FINISHED, it would run MODULE_B. If its status was not equal to FINISHED, it would run MODULE_C.

- Creating a chain and adding the three modules (single threaded in alphabetic order). Then, adding a CHECK HISTORY condition to MODULE_B so that if it was not successful, its action would be SKIP TASK. And adding a CHECK HISTORY condition to MODULE_C so that if it was successful, its action would be SKIP TASK.

Note: These methods do not take the virtual workday into consideration.
12.8 Example: Running on the Successful Completion of a Chain Component

Predecessors allow you to run a series of chain components in a particular order without creating multiple chains.

There may be times when you want to run a series of modules in a chain in a particular order. For example, after program “A” runs, you want programs “B” and “C” to run. If program “C” completes successfully, you want program “D” to run. The logic is illustrated in Figure A.

For this example, you could create a chain and add four modules to it:

- **MODULE_A**, which runs program “A”. You would not check the Multi thread box for this component so the other components will have to wait to run until it completes.
- **MODULE_B**, which runs program “B”. You would check the Multi thread box for this component. This way it can run concurrently with either **MODULE_C** or **MODULE_D**.
- **MODULE_C**, which runs program “C”. You would check the Multi thread box for this component. This way it can run concurrently with **MODULE_B**.
- **MODULE_D**, which runs program “D”. You would check the Multi thread box for this component. This way it can run concurrently with **MODULE_B**. On this component you would add the following predecessor statement to allow it to run once **MODULE_C** finishes successfully.

\[ \text{MODULE}_C = \text{Success} \]

To define this predecessor, you would select the Chain Components sub-tab, select the chain name in the Chain field, and choose the component (see Figure B).

Without using a predecessor statement, you would have to create a second chain that includes **MODULE_C** and **MODULE_D** as single threaded modules and add that chain to the first chain.
Figure B. To have a predecessor check for a chain component, use the Chain Components sub-tab.
# 13 Working with Conditions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>Introduction to Working with Conditions</td>
<td>300</td>
</tr>
<tr>
<td>13.2</td>
<td>How AppWorx Processes Conditions</td>
<td>302</td>
</tr>
<tr>
<td>13.3</td>
<td>Adding, Editing, and Deleting Conditions</td>
<td>304</td>
</tr>
<tr>
<td>13.4</td>
<td>Selecting Condition Values</td>
<td>306</td>
</tr>
<tr>
<td>13.4.1</td>
<td>Selecting Condition Timing</td>
<td>308</td>
</tr>
<tr>
<td>13.4.2</td>
<td>Selecting Condition Types</td>
<td>310</td>
</tr>
<tr>
<td>13.4.3</td>
<td>Selecting Condition Qualifiers</td>
<td>312</td>
</tr>
<tr>
<td>13.4.4</td>
<td>Selecting Condition Actions</td>
<td>314</td>
</tr>
<tr>
<td>13.4.5</td>
<td>Selecting Action Timing</td>
<td>316</td>
</tr>
<tr>
<td>13.4.6</td>
<td>Adding Substitution Variables and Replacement Values to Conditions</td>
<td>318</td>
</tr>
<tr>
<td>13.4.7</td>
<td>Using Replacement Values in Conditions</td>
<td>320</td>
</tr>
<tr>
<td>13.5</td>
<td>Using Conditions to Accomplish Common Tasks</td>
<td>322</td>
</tr>
<tr>
<td>13.5.1</td>
<td>Checking on Time Since Request</td>
<td>324</td>
</tr>
<tr>
<td>13.5.2</td>
<td>Checking if a File Exists</td>
<td>326</td>
</tr>
<tr>
<td>13.5.3</td>
<td>Checking if a Process Is Running</td>
<td>328</td>
</tr>
<tr>
<td>13.5.4</td>
<td>Checking for a Finished Job in the History</td>
<td>330</td>
</tr>
<tr>
<td>13.5.5</td>
<td>Killing a Job That is Running Too Long</td>
<td>332</td>
</tr>
<tr>
<td>13.5.6</td>
<td>Requesting a Module</td>
<td>334</td>
</tr>
<tr>
<td>13.5.7</td>
<td>Triggering Two or More Actions on the Same Event</td>
<td>336</td>
</tr>
<tr>
<td>13.5.8</td>
<td>Running One Module Based on the Failure of Another</td>
<td>338</td>
</tr>
<tr>
<td>13.5.9</td>
<td>Controlling Execution Sequence in a Chain</td>
<td>340</td>
</tr>
</tbody>
</table>
13.1 Introduction to Working with Conditions

Conditions control the execution of modules and chains, they can apply to modules, chains, or chain components. Conditions can be evaluated before, during, and after a job executes, or after a job is deleted. You can add, update, or delete conditions for the particular running of a job from the Backlog.

Conditions are powerful tools for controlling the execution of jobs. A module with two conditions is shown in Figure A.

Using conditions, you can do such things as:

- Check the current time and put a job on hold if it is later than 5 A.M.
- Check if a file exists, and if it does not, wait 15 minutes and check again.
- Run the second module if the first module in a chain completes successfully. Run the third module if the first module aborts.

Conditions can apply to modules, chains, or chain components. By adding conditions to chain components, you can use a module differently in several chains, or even the same chain. You can also add, edit, or delete conditions for non-running jobs in the Backlog. When you do this, the changes are particular only to that running of the job.

AppWorx can evaluate the conditions assigned to a module before, during, and after the module executes, and after a module is deleted, depending on the timing in the condition’s definition. Conditions are not evaluated until the module is submitted and displayed in the Backlog. This includes conditions that are evaluated before a module executes. This means all components in a chain are displayed in the Backlog regardless of the conditions assigned to them.

When a test defined in a condition is met, AppWorx takes the action defined the condition. AppWorx includes a number of actions. If you want to take more than one action based on an event, you create multiple conditions, each with the same test but different actions.
Using Your Corporate Database to Control Operations

One of the more powerful features of AppWorx is the ability to control operations based on the content of your corporate database. By using dynamic substitution variables in conditions, you can make decisions based on such data as:

- The number of records in a table
- The total dollars represented by receivables
- The date a database was last updated

For information on including substitution variables in conditions, see topic 13.4.6 Adding Substitution Variables and Replacement Values to Conditions.

Using Module/Chain Conditions for Chain Components

You can use the conditions defined for modules or chains when adding them to a chain. To do this, select the Use module conditions option for the chain components. For more information, see topic 9.4.2 Setting Component Execution Options

The USER DEFINED Condition

To enter a text string, substitution variable, or replacement value as your condition values, select the USER DEFINED condition. You can use the USER DEFINED condition to take an action every time a module is run by entering 1=1 in the Test box as shown in Figure B.

Bad Conditions

If AppWorx evaluates a bad condition (for example a condition with a bad SQL statement), it will ignore the condition and report it to the master’s log file. The module’s status will be set to BAD CONDITN.

Module Conflicts

The module conflicts feature existed in prior versions of AppWorx. With module conflicts, you could prevent a module from starting if a particular job was running in the Backlog. In AppWorx 5.1, conditions replace module conflicts. Module conflicts included in your modules’ definitions are automatically translated to conditions during the upgrading process.
13.2 How AppWorx Processes Conditions

AppWorx processes conditions before, during, and after a module runs, and after a job is deleted.

The flowchart in Figure A shows the evaluation cycle AppWorx applies to each job’s conditions. An evaluation cycle consists of working through the flowchart in Figure A until the ‘Run job’ or ‘End processing’ step is reached. AppWorx runs through the evaluation cycle for a job until it completes executing and is moved to the History, or until it aborts. AppWorx evaluates all conditions in the order they are listed with the following exceptions:

- If a condition runs a job, AppWorx ignores the remaining BEFORE conditions for that job.
- If a condition stops the evaluation cycle, all remaining conditions are ignored for the current cycle. For example if a condition puts a module on hold, all remaining conditions will be ignored until the next cycle.
- If you use the GOTO CONDITION action, the destination condition must come after the condition containing the GOTO action.

Running BEFORE Conditions

When AppWorx is ready to execute a job, it evaluates its BEFORE conditions. By default, a job will run unless one of the BEFORE conditions modifies its eligibility.

Running DURING Conditions

While a job is running, AppWorx evaluates its DURING conditions. The frequency that AppWorx evaluates DURING conditions is based on the DURING_WAIT variable setting.

When the master has no processing to do, it sleeps for the number of seconds entered in the Sleep time field for its local agent. At the end of each sleep cycle, the master evaluates the DURING conditions.

When the master is processing, it checks the DURING conditions. The frequency that it checks them is based on the setting for the DURING_WAIT variable in your awenv.ini file, located in the site folder. You can modify the setting; the default is 60 seconds.

Setting a low DURING_WAIT value may impede performance without evaluating conditions more frequently.

If a job completes executing in a shorter amount of time than the setting for the DURING_WAIT variable, its DURING conditions may not execute.

Running AFTER Conditions

After a job completes executing successfully or unsuccessfully, AppWorx evaluates all AFTER conditions. If a job is killed, AFTER conditions are still evaluated.
Running DELETED Conditions

If a job is deleted by a user, AppWorx processes DELETED conditions and skips any AFTER conditions associated with the given module.

Start

Is there a condition that has not been evaluated?

Yes

Is the condition true?

Yes

Take action and set 'action taken' flag.

Yes

Does the action stop processing?

Yes

End processing

No

Run job

No

Is the condition true?

Yes

Take action and set 'action taken' flag.

Yes

Does the action stop processing?

Yes

End processing

No

Figure A. How AppWorx processes conditions
13.3 Adding, Editing, and Deleting Conditions

To add, edit, or delete conditions for a module, chain, or chain component, select that object’s Conditions tab and click New, Edit, or Delete. To change the order of a condition, click Up or Down. You can also add, edit, or delete conditions for non-running jobs in the Backlog. When you do this, the changes are particular only to that running of the job.

You can include conditions for modules, chains, or chain components. The Conditions tab for a module is shown in Figure A.

**Figure A.** The Conditions tab displays the conditions for a module.

**Adding Conditions**

To define add a condition to a module, chain, or chain detail:

1. From the Conditions tab for a module, or chain, or chain component, click New. AppWorx displays the Condition Details window shown in Figure B.

**Figure B.** The Conditions Detail window.
2. Complete the fields on the Condition Details window as appropriate.
   For more information on completing the fields, see the following topics in this chapter.
3. To save the module/chain with the new condition, click **OK**.
   AppWorx adds the condition and automatically saves the module/chain definition.
   **Note:** If you decide you do not want to include the new condition in the module/chain definition, you must delete the condition. If you click **Cancel** on the Modules or Chains window after adding a condition, the condition is still included.

**Editing and Deleting Conditions**

To edit or delete an existing condition for a module, select the module on the Conditions tab and click **Edit** or **Delete**.

**Setting Conditions Order**

To change the order of the conditions for a module on the Conditions tab, select the condition you want to move and click **Up** or **Down**. Conditions must remain with their timing group (BEFORE, DURING, AFTER, or DELETED).

**Taking More Than One Action for an Event**

You can define only one action for each condition. If you want to take more than one action based on the same event, create multiple conditions with different actions and set the condition order on the Condition tab. If one condition takes an action that cancels the task or chain, be sure it is the last one in the related group. For an example, see topic 13.5.7 **Triggering Two or More Actions on the Same Event**.

**Killing All Components in a Chain**

Selecting a KILL TASK action for a DURING condition on a chain will kill all running components of the chain.
13.4 Selecting Condition Values

The conditions in the Select Conditions drop-down box are based on the selection you make in the Timing field. The fields displayed in the Test box are based on the condition you select.

There are several sources of values for the conditions:

- You can select a value from a field with a drop-down box.
- You can enter a text string where allowed.
- You can select a replacement value or substitution variable for any field that includes the ... button. Replacement values retrieve values stored for AppWorx objects, while substitution variables represent one of many possible dynamic or static values stored in any designated database.
- If the USER DEFINED condition is selected, you can enter a numbered substitution variable (#1 - #99) to accept a value passed down from a chain when it is requested on an ad hoc basis. For more information on passing values through a chain, see topic 11.10 Passing Values Through a Chain with Prompts.
Subtopics

The values available for conditions are described in the subtopics that follow:

13.4.1 Selecting Condition Timing
13.4.2 Selecting Condition Types
13.4.3 Selecting Condition Qualifiers
13.4.4 Selecting Condition Actions
13.4.5 Selecting Action Timing
13.4.6 Adding Substitution Variables and Replacement Values to Conditions
13.4.7 Using Replacement Values in Conditions
13.4 Selecting Condition Values

13.4.1 Selecting Condition Timing

You can set the timing for a condition to have AppWorx check it before, during, or after a module or chain executes, or after it deletes. The option selected from the Timing drop-down box determines which conditions are available.

You can choose to have a condition checked before, during, or after a module or chain executes, or after it deletes.

![Condition Details Window](image)

*Figure A. Timing lets you control when AppWorx evaluates a condition.*

The **Timing** field in the Condition Details window lets you control when AppWorx checks for a condition, and which conditions are available. The timing options are described below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEFORE</td>
<td>The condition is evaluated before the module/chain executes.</td>
</tr>
<tr>
<td>DURING</td>
<td>The condition is evaluated every time the master checks the Backlog queue (about once a minute) while the module/chain is executing.</td>
</tr>
<tr>
<td>AFTER</td>
<td>The condition is evaluated after the module/chain executes, even if it aborts or is killed.</td>
</tr>
<tr>
<td>DELETED</td>
<td>If a module/chain is DELETED, the conditions on the DELETED module will be evaluated. This differs from an AFTER condition. AFTER conditions are not evaluated for a DELETED job.</td>
</tr>
</tbody>
</table>
13.4 Selecting Condition Values

13.4.2 Selecting Condition Types

The values available for the Condition field are described below.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Timing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER DEFINED</td>
<td>BDAR</td>
<td>Provides two Test fields, separated by a qualifier. You define the condition by entering text strings or numeral substitution variables, or use the ... button to select substitution variables/replacement values. You can use the USER DEFINED condition to take an action every time a module is run by entering 1=1 in the Test box.</td>
</tr>
<tr>
<td>CHECK FILE*</td>
<td>BDA</td>
<td>Checks if the file named in the File Name field does or does not exist based on maximum file age, minimum unchanged time (stability), and/or minimum file size on the agent where the module is running. Wildcards and environment variables are allowed if the file name is enclosed in double quotes. The first line of the last file found by CHECK FILE is stored in the replacement value {file-line}. You might use the {file-line} replacement value in a later condition to check whether the file included a text based error code and set the job's status accordingly if it did. For a CHECK FILE example, see topic 13.5.2 Checking if a File Exists.</td>
</tr>
<tr>
<td>CHECK HISTORY*</td>
<td>BDA</td>
<td>Checks for the status of a module in the History within the last days, hours, and minutes. Use the Alias when referring to chain components. The condition is evaluated as true if: • Success is selected and the module has a status of FINISHED. -or- • Failure is selected and the module has a status of DIED, ABORTED, KILLED, or TIMEDOUT. If the module is not found, the condition is evaluated as false. For an example, see topic 13.5.4 Checking for a Finished Job in the History.</td>
</tr>
<tr>
<td>Condition</td>
<td>Timing</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CHECK PROCESS</td>
<td>BDA</td>
<td>Checks if another process is running on the agent where the module is running. For an example, see topic 13.5.3 Checking if a Process Is Running.</td>
</tr>
<tr>
<td>CURRENT QUEUE*</td>
<td>BR</td>
<td>Provides the value for the current queue definition.</td>
</tr>
<tr>
<td>CURRENT TIME*</td>
<td>BDAR</td>
<td>The current time in the format HH:MM:SS.</td>
</tr>
<tr>
<td>MODULE REQUESTED*</td>
<td>B</td>
<td>Checks if a module has been requested and is in the Backlog.</td>
</tr>
<tr>
<td>MODULE RUNNING*</td>
<td>B</td>
<td>Checks if a module is in a running state in the Backlog. For an example, see topic 13.5.5 Killing a Job That is Running Too Long.</td>
</tr>
<tr>
<td>RETURN CODE*</td>
<td>A</td>
<td>Reads the return code generated by the program.</td>
</tr>
<tr>
<td>RUN TIME*</td>
<td>D</td>
<td>The wall clock time since the module started running.</td>
</tr>
<tr>
<td>STATUS*</td>
<td>A</td>
<td>Usually used to check whether a job’s status is equal to FINISHED. This allows you to check if the current module completed successfully.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For a complete list of statuses, see Appendix B: Job Status Values. For an example, see topic 13.5.5 Killing a Job That is Running Too Long.</td>
</tr>
<tr>
<td>TIME SINCE REQUEST*</td>
<td>BDAR</td>
<td>Elapsed wall clock time between the time the module was requested and the current time. Useful for checking how long a job has been waiting in the Backlog. For an example, see topic 13.5.1 Checking on Time Since Request.</td>
</tr>
</tbody>
</table>

*System function.
13.4 Selecting Condition Values

13.4.3 Selecting Condition Qualifiers

Qualifier values are available based on the condition you select. You can pick from string and numeric qualifiers.

Qualifiers are used to evaluate two values in the Test box of a condition (see Figure A). The condition you select determines the qualifiers that are available from the drop-down box.

**Qualifier Types**

The qualifier you select determines the way the condition is evaluated. The two types of qualifiers are described in the table below:

<table>
<thead>
<tr>
<th>Qualifier type</th>
<th>Evaluate the condition values as:</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>A left to right sequence of characters.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Mathematical statements.</td>
</tr>
</tbody>
</table>

If you select the wrong type of qualifier, AppWorx may not evaluate your condition the way you intend it to. For example, AppWorx evaluates ‘1000 GT 50’ as true (because one thousand is more than fifty), and ‘1000 > 50’ as false (because 1 does not come after 5 alphabetically).
Qualifier Descriptions

Qualifier descriptions are listed in the table below.

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!= or &lt;&gt;</td>
<td>String</td>
<td>Not equal–both signs evaluate the same.</td>
</tr>
<tr>
<td>&lt;</td>
<td>String</td>
<td>Less than.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>String</td>
<td>Less than or equal to.</td>
</tr>
<tr>
<td>=</td>
<td>String</td>
<td>Equal to.</td>
</tr>
<tr>
<td>&gt;</td>
<td>String</td>
<td>Greater than.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>String</td>
<td>Greater than or equal to.</td>
</tr>
<tr>
<td>LK</td>
<td>String</td>
<td>Is like–uses a substring search to check if the second value occurs in the first. You might use this qualifier to determine whether a substring exists in a dynamic substitution variable's value.</td>
</tr>
<tr>
<td>GE</td>
<td>Numeric</td>
<td>Greater than or equal to.</td>
</tr>
<tr>
<td>LE</td>
<td>Numeric</td>
<td>Less than or equal to.</td>
</tr>
<tr>
<td>GT</td>
<td>Numeric</td>
<td>Greater than.</td>
</tr>
<tr>
<td>LT</td>
<td>Numeric</td>
<td>Less than.</td>
</tr>
<tr>
<td>NE</td>
<td>Numeric</td>
<td>Not equal.</td>
</tr>
<tr>
<td>EQ</td>
<td>Numeric</td>
<td>Equal to.</td>
</tr>
</tbody>
</table>
When a condition is true, AppWorx can take a variety of actions. Some of the actions require that you enter additional information.

The values available for the **Action** field are listed in the table below. The action values are determined by the selected **Timing** option. In the table the letters in the Timing column stand for the following: B–Before, D–During, A–After, and R–Delete.

<table>
<thead>
<tr>
<th>Action</th>
<th>Timing</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT TASK*</td>
<td>BA</td>
<td>Aborts the current module. Also see the SET ABORT STATUS action.</td>
</tr>
<tr>
<td>CANCEL CHAIN*</td>
<td>BDA</td>
<td>Cancels the chain and all of its components that are not running.</td>
</tr>
<tr>
<td>CHANGE Q*</td>
<td>BD</td>
<td>Changes the queue for this job. Changing queues does not affect the chain execution order. The <strong>The first time the condition is true</strong> radio button must be selected for this action or the job will never start. Changing queues does not affect the chain execution order. The <strong>The first time the condition is true</strong> radio button must be selected for this action or the job will never start. Switching the queue in the Backlog will not disengage a component from its chain. If you wish to disengage a component from its chain, you can do so from the chain’s definition. On the chain’s Components tab go to the <strong>Link</strong> field and select “Unlink on start”. For more information, see topic <strong>9.4.2 Setting Component Execution Options</strong>.</td>
</tr>
<tr>
<td>DELAY TASK*</td>
<td>B</td>
<td>Delays for a number of hours, minutes, and seconds before the job is run. HH:MM:SS format. The time delay begins when the condition is checked.</td>
</tr>
<tr>
<td>DELETE SUBVAR</td>
<td>BDAR</td>
<td>Deletes a substitution variable (or substitution variables when wildcards are used).</td>
</tr>
<tr>
<td>GOTO CONDITION</td>
<td>BDA</td>
<td>Skips to a condition by number. Destination condition must follow the condition containing the GOTO action. <strong>Note:</strong> The GOTO number is not re-sequenced when a condition is moved.</td>
</tr>
<tr>
<td>HOLD TASK*</td>
<td>B</td>
<td>Changes the module’s status to HOLD. Also see the SET HOLD STATUS action.</td>
</tr>
<tr>
<td>KILL TASK*</td>
<td>D</td>
<td>Changes the module’s status to KILL.</td>
</tr>
<tr>
<td>REQUEST MODULE</td>
<td>BDAR</td>
<td>Select the <strong>Details</strong> button to select a module and set its options. For an example, see topic <strong>13.5.6 Requesting a Module</strong>.</td>
</tr>
<tr>
<td>Action</td>
<td>Timing</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RESCHEDULE TASK*</td>
<td>B</td>
<td>The time of day to reschedule the task today. HH:MM:SS format. If a time earlier than current is entered, the module will run as soon as possible.</td>
</tr>
<tr>
<td>RESTART ON ABORT</td>
<td>BDA</td>
<td>Set or reset the module’s restart on abort attribute. This action does not alter the module definition—it affects the particular module in the particular chain.</td>
</tr>
<tr>
<td>RUN HOST COMMAND</td>
<td>DAR</td>
<td>Runs an operating system host command with arguments on the agent of the module (for example <code>cp file1 file2</code>). For more information, see your OS documentation.</td>
</tr>
<tr>
<td>RUN TASK*</td>
<td>B</td>
<td>Runs the task. Any remaining ‘Before’ conditions are ignored.</td>
</tr>
<tr>
<td>SET ABORT STATUS*</td>
<td>A</td>
<td>Sets module status to ABORTED and displays custom text as status name. Allows for a maximum of 12 characters.</td>
</tr>
<tr>
<td>SET FINISHED STATUS*</td>
<td>A</td>
<td>Set module status to FINISHED and display custom text as status name. Allows for a maximum of 12 characters.</td>
</tr>
<tr>
<td>SET HOLD STATUS*</td>
<td>B</td>
<td>Sets module status to HOLD and displays custom text as status name. Allows for a maximum of 12 characters.</td>
</tr>
<tr>
<td>SET SKIP STATUS*</td>
<td>B</td>
<td>Sets module status to SKIPPED and displays custom text as status name. Allows for a maximum of 12 characters.</td>
</tr>
<tr>
<td>SET SUBVAR BDAR</td>
<td>DAR</td>
<td>Creates a static substitution variable. To create a unique substitution variable for a chain, include the {chain_id} or {chain_seq} replacement value. To create a unique substitution variable for a module, include the {jobid} replacement value. For a description of replacement values, see topic 13.4.7 Using Replacement Values in Conditions.</td>
</tr>
<tr>
<td>SKIP TASK*</td>
<td>B</td>
<td>Sets the module’s status to SKIPPED. Also see the SET SKIP STATUS action.</td>
</tr>
<tr>
<td>STAY IN QUEUE ON ABORT</td>
<td>BDA</td>
<td>Sets or resets the module’s Stay in queue on abort attribute. This action does not alter the module definition—it affects the run of the particular module in the particular chain.</td>
</tr>
<tr>
<td>WAIT UNTIL MET*</td>
<td>B</td>
<td>Causes the job to pause until the condition is met.</td>
</tr>
</tbody>
</table>

*Stops processing module conditions for the current evaluation cycle and will not start the job this evaluation cycle.*
13.4  Selecting Condition Values

13.4.5  Selecting Action Timing

You determine if the action will be taken the first time the condition is true, every time the condition is true, or if the condition should be disabled.

AppWorx evaluates conditions for a module at least every 59 seconds. You select how often the action for each condition should be taken by selecting a radio button from the Action Timing box as shown in Figure A.

![Figure A. Select an action timing for a condition.](image)

The action timing radio buttons are described in the table below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first time the condition is true</td>
<td>AppWorx only initiates the specified action the first time the condition is true and changes its action timing to Disabled. After that, AppWorx skips the condition. You might use this option for a condition that notifies an operator when a module has failed. If there are events that cause the condition to be evaluated as true a number of times, AppWorx won’t send out a new message every time.</td>
</tr>
<tr>
<td>Every time the condition is true</td>
<td>AppWorx initiates the specified action every time the condition is true. You might use this option if you want module ‘Y’ to execute only after another module ‘X’ has completed. You could check for the status of module ‘X’, and if it has not completed, delay module ‘Y’ by 15 minutes. You would want to use the delay each time the condition is true.</td>
</tr>
</tbody>
</table>
Disabled AppWorx will skip the condition.
You might use this option if you write a condition, but do not
want to activate it.

**Note:** The action timing for conditions with **The first time the condition is true** selected will change to **Disabled** in the
Backlog/History for the running of a job when the action is
taken.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disabled</td>
<td>AppWorx will skip the condition. You might use this option if you write a condition, but do not want to activate it. <strong>Note:</strong> The action timing for conditions with <strong>The first time the condition is true</strong> selected will change to <strong>Disabled</strong> in the Backlog/History for the running of a job when the action is taken.</td>
</tr>
</tbody>
</table>
13.4 Selecting Condition Values

13.4.6 Adding Substitution Variables and Replacement Values to Conditions

You can add a substitution variable or replacement variable to a field using the Subvar Replacement Values window.

Substitution variables let you store values that can be referenced in modules and chains. The values can be stored in a database table or generated by a SQL statement at the time a job is submitted. For more information on substitution variables, see chapter 14: Defining Substitution Variables.

Replacement values represent values assigned to AppWorx objects that are stored in the AppWorx database. They come predefined when you install AppWorx. For a list of replacement values, see topic 13.4.7 Using Replacement Values in Conditions.

Procedure

To add a substitution variable or replacement variable to a field with the Subvar Replacement Values window.

1. Click the ... button next to a field, or press F2 when the field is selected.
   AppWorx displays the Subvar Replacement Values window shown in Figure A.

2. Select one value from either list.
   You can type the first letter of a value to highlight it.

3. Click OK to add the value to a field.

Note: If the cursor is in the middle of a text string when you bring up the Subvar Replacement Values window, AppWorx will place the value at the cursor’s position.

You can enter substitution variables/replacement values by typing the value into the field. Substitution variables are preceded by a ‘#’; replacement values are written inside curly braces, ‘{ }’.
Substitution Variables Evaluated as Characters or Numbers

AppWorx evaluates substitution variables as characters or numbers depending on the qualifier value used in the Condition Details window. For example, the condition #chain = bad will be evaluated as a string. The condition #count_employees LE 100 will be evaluated numerically. For a complete list of string and numeric qualifiers, see topic 13.4.3 Selecting Condition Qualifiers.

To ensure proper evaluation of dates, use YYYYMMDD format with string qualifiers (>, <=, =) or Julian dates (dates expressed as the number of days elapsed since January 1, 4713 B.C.) with numeric qualifiers (for example GE, LE).
13.4 Selecting Condition Values

13.4.7 Using Replacement Values in Conditions

Use replacement values to retrieve values assigned to an AppWorx object and stored in the AppWorx database. They come predefined when you install AppWorx.

Replacement values retrieve values stored for AppWorx objects. They come predefined when you install AppWorx. The replacement values are described in Table A. You can include replacement values in a condition using the Subvar Replacement Values window. For details on the Subvar Replacement Values window, see topic 13.4.6 Adding Substitution Variables and Replacement Values to Conditions.

In Figure A, if the module does not complete successfully, the {condition1} replacement value is used to set the value of the #job_status substitution variable to the status of the module. This substitution variable can then be evaluated in other conditions.

Table A. Replacement values

<table>
<thead>
<tr>
<th>Replacement value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{agent}</td>
<td>The name of the agent that the job is running under. <strong>Note:</strong> If a condition calls a module that uses the {agent} variable, {agent} refers to the parent module’s agent.</td>
</tr>
<tr>
<td>{application_path}</td>
<td>The library assigned to the module.</td>
</tr>
<tr>
<td>{application}</td>
<td>The application assigned to the module.</td>
</tr>
<tr>
<td>{batch_queue}</td>
<td>The current queue assigned to the job in the Backlog.</td>
</tr>
<tr>
<td>{chain_id}</td>
<td>The number assigned to a chain at the time the chain is executed. Used to create a unique substitution variable each time a chain is run. Example: #value{chain_id}=success. For more information, see topic 14.6 Creating Local Substitution Variables Using Replacement Values.</td>
</tr>
<tr>
<td>{chain_seq}</td>
<td>The chain sequence number assigned to a chain when it is created.</td>
</tr>
<tr>
<td>Replacement value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>{command_path}</code></td>
<td>The subdirectory specified for the program type assigned to the module. The subdirectory is specified in the <strong>Command path</strong> field in the Program Type window.</td>
</tr>
<tr>
<td><code>{command_type}</code></td>
<td>The program type assigned to the module.</td>
</tr>
<tr>
<td><code>{command}</code></td>
<td>The name of the program run by the module. The program name is selected when defining the module.</td>
</tr>
<tr>
<td><code>{condition1}</code></td>
<td>If STATUS is entered in the first field of a USER DEFINED condition, this replacement status value will read the value of the job.</td>
</tr>
<tr>
<td><code>{condition2}</code></td>
<td>If STATUS is entered in the second field of a USER DEFINED condition, this replacement status value will read the value of the job.</td>
</tr>
</tbody>
</table>
| `{db_type}`       | The database type for the login assigned to the module.  
**Note:** This value is always Oracle now. |
| `{file_line}`     | The first line of the last file found by CHECK FILE. It remains available until a CHECK FILE condition is run for a different file. |
| `{host_command}`  | The name of the program type script run by the program type assigned to the module. For example: SQLP. The scripts are usually located in the **exec** subdirectory. The script is identified in the **Host command** field in the Program Type window. |
| `{jobid}`         | A unique number assigned to a job by AppWorx at the time the job runs. |
| `{master}`        | The name of the master. |
| `{mjn}`           | The module sequence number assigned to the module when it is created. |
| `{module}`        | The module name or alias assigned to a module. |
| `{net_connect}`   | The connect string associated with the login assigned to the module. The connect string is specified by the AppWorx login. |
| `{node}`          | Gives the name of the agent or agent group as defined for the module.  
**Note:** Use `{agent}` to view the name of the agent that the job is running under. |
| `{operator}`      | Refers to the agent a job will run on. This replacement value has been replaced by `{agent}`. It remains for legacy customers who use it. |
| `{oracle_sid}`    | The login associated with the database assigned to the module. The database ID is specified by the AppWorx login. |
| `{requestor}`     | The user ID assigned to the module if it is submitted on an ad hoc basis |
13.5 Using Conditions to Accomplish Common Tasks

You can accomplish common types of scheduling tasks by adding conditions to modules in chains.

The previous topics in this chapter described how AppWorx evaluates conditions and the values available for them. The topics that follow give examples of how they can be used in various modules, chains, and chain components.

Subtopics

Conditions are one of the more complex areas of AppWorx. To help you understand how they can be used, several common scheduling tasks are described in the subtopics that follow:

13.5.1 Checking on Time Since Request
13.5.2 Checking if a File Exists
13.5.3 Checking if a Process Is Running
13.5.4 Checking for a Finished Job in the History
13.5.5 Killing a Job That is Running Too Long
13.5.6 Requesting a Module
13.5.7 Triggering Two or More Actions on the Same Event
13.5.8 Running One Module Based on the Failure of Another
13.5.9 Controlling Execution Sequence in a Chain
13.5 Using Conditions to Accomplish Common Tasks

13.5.1 Checking on Time Since Request

You may want to abort a job if it does not run within a certain amount of time after it is submitted. You can do this using the TIME SINCE REQUEST condition with an ABORT TASK action.

If you have scheduled a job to run at a specific time, you may want to set up a condition that aborts the job if it has been waiting in the Backlog too long.

The settings shown in Figure A abort the job if it has been waiting more than 30 minutes. The Timing is set to BEFORE, so the check will be made before the module is executed. The first time the condition is true is selected because the ABORT TASK action only needs to be taken once.

![Figure A](image.png)

**Figure A.** This condition cancels the chain if this job has not run 30 minutes after it was requested.

Related Topic

You can abort a stand-alone module if it has run too long is to enter a value in the Max run time field on a module’s General tab. For more information, see topic 8.4.6 Entering Execution Options.
13.5 Using Conditions to Accomplish Common Tasks

13.5.2 Checking if a File Exists

Some modules need to wait for the creation of one or more files before executing. You can check for the existence of a file (considering age and size) using the CHECK FILE condition.

If a module in a chain needs to wait for the creation of a file before executing, you can delay the module until the necessary output file is available.

Many times, however, a check for a simple file is not enough. The required file may need to be a certain size, or the file may need to have been created within the last 24 hours to be valid. In addition to checking for a file by file name, you can check for files using environment variables and wildcards, specifying maximum acceptable age or minimum acceptable stability (of the file based upon the last modification date/time of the file), and specifying the minimum acceptable size.

When using wildcards, only one file needs to match the wildcard, age, size, and stability requirements for the condition to evaluate to TRUE.

The first line of the last file found by CHECK FILE is stored in the replacement value {file-line}. You might use the {file-line} replacement value in a later condition to check whether the file included a particular text based error code and set the job’s status accordingly if it did.

Note: CHECK FILE and CHECK PROCESS are the only conditions that execute actions when they are evaluated as false.

![Figure A](image.png)

*Figure A. This condition checks whether a file exists, it will wait until it does before running the job.*
A sample CHECK FILE condition is shown in Figure A. **Every time the condition is true** is selected, so the condition will be checked every evaluation cycle, even after it has waited once.

The condition in Figure A will wait until it finds any single file in the $AW_HOME/out directory that has a rep extension, has not changed for at least 180 seconds, is less than 1 day old, and is larger than 10,000 bytes.

**Field Descriptions**

The CHECK FILE fields are described below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>The name of the file to check. You can use environment variables and wildcards to specify the name, but the name should be enclosed in single quotes if wildcards are used as shown in Figure A.</td>
</tr>
<tr>
<td>Exists/ Does Not Exist</td>
<td>Determines whether to check if a file exists or does not exist. Choosing Exists will allow the use of the fields listed below. Choosing Does Not Exist will only allow the use of File Name field.</td>
</tr>
<tr>
<td>Days Hours Minutes</td>
<td>These fields can be used to specify a maximum acceptable age for the file. The fields will accept values up to 99. If the last modification date/time of the file is older than this setting, the file is considered nonexistent and the condition will evaluate as false.</td>
</tr>
<tr>
<td>And has not changed for _____ seconds</td>
<td>Specify a minimum time (in seconds) for which the file must have been stable (unchanged). The field will accept values up to 999. If the file has been modified more recently than this many seconds ago, the file is not stable. An unstable file is considered nonexistent and the condition will evaluate as false.</td>
</tr>
<tr>
<td>And has a minimum size of _____ bytes</td>
<td>Specify a minimum acceptable size (in bytes) for the file. The field will accept values up to 9999999 (10 megabytes). If the file is smaller than this setting, the file is considered nonexistent and the condition will evaluate as false.</td>
</tr>
</tbody>
</table>
13.5 Using Conditions to Accomplish Common Tasks

13.5.3 Checking if a Process Is Running

Some modules need to wait for another program or application to be running before executing. You can check if a program is processing using the CHECK PROCESS condition.

Some modules need to have another process running before they will start. This may include checking Oracle PMON, SMON, or DBWR processes. You can check if a program is running using the CHECK PROCESS condition. You can check for the process’s existence and take appropriate actions depending on the results.

CHECK PROCESS is executed on the agent where the module is running or is going to run.

Note: If a module or a chain should not run while a backup is executing, a condition should be defined to check if the backup process is still running, and wait until it finishes.

A sample CHECK PROCESS condition is shown in Figure A. Every time the condition is true is selected because the WAIT UNTIL MET action needs to be taken each time AppWorx processes the transaction.

![Figure A](image)

*Figure A. This condition checks whether a process is running, it will wait until it is before running the job.*
13.5 Using Conditions to Accomplish Common Tasks

13.5.4 Checking for a Finished Job in the History

The CHECK HISTORY condition checks for the existence of a FINISHED or non-FINISHED job in the History within a given time period.

The CHECK HISTORY condition can be used to check the History for the most recent existence of a job with a success (FINISHED) or failure (non-FINISHED) status within a given time period of days, hours and minutes. The time period is based on the finish date and time of the most recent job.

A sample CHECK HISTORY condition is shown in Figure A. Every time the condition is true is selected because the WAIT UNTIL MET action needs to be taken each time AppWorx processes the transaction.

![Figure A. This condition checks whether a module has finished, it will wait until it has before running this job.](image)

The condition in Figure A will wait until it finds an FTP_TRANSFER job that FINISHED in the last 45 minutes.
Field Descriptions

The CHECK HISTORY fields are described below:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module/View Name</td>
<td>The module name or alias of the (most recent) job to check.</td>
</tr>
<tr>
<td>Success/Failure</td>
<td>Determines whether the check is made for a successful job or a failed job.</td>
</tr>
<tr>
<td></td>
<td>Successful is defined as a job with a status of FINISHED; failed is defined</td>
</tr>
<tr>
<td></td>
<td>as a job with any completion status other than FINISHED (such as DIED,</td>
</tr>
<tr>
<td></td>
<td>ABORTED, or TIMED OUT).</td>
</tr>
<tr>
<td>Days Hours Minutes</td>
<td>These fields can be used to specify a maximum acceptable age for the job.</td>
</tr>
<tr>
<td></td>
<td>The fields will accept values up to 99.</td>
</tr>
<tr>
<td></td>
<td>If the finish date and time of the job is older than this setting, the</td>
</tr>
<tr>
<td></td>
<td>condition will evaluate to false.</td>
</tr>
</tbody>
</table>

CHECK HISTORY is available as a BEFORE, DURING and AFTER condition. Any of the available actions may be used with CHECK HISTORY.
A module that is taking too long to execute often indicates a problem. You can use the RUN TIME condition to check on a module and issue an abort command if it has run longer than a specified time.

You generally know how long it should take a module to complete executing. Some modules may take only a few minutes, while others may take several hours. If a module has been executing for longer than expected, it usually indicates a problem. You can use a condition to check on how long a module has been running and cancel the module if it has run longer than expected.

The settings shown in Figure A check to see if the run time is greater than 30 minutes. If it is, AppWorx kills the module.

The first time the condition is true is selected, because the module only needs to be killed one time.

The RUN TIME condition is automatically checked for running tasks. Its Timing must be set to DURING.

![Figure A. This condition will abort the job if it has been running for longer than 30 minutes.]

**Related Topic**

You can abort a stand-alone module if it has run too long is to enter a value in the Max run time field on a module’s General tab. For more information, see topic 8.4.6 Entering Execution Options.
13.5 Using Conditions to Accomplish Common Tasks

13.5.6 Requesting a Module

There may be times when you wish to call a module or chain as a condition’s action. You can use the REQUEST MODULE action to select options and call a module or chain with a condition.

There may be times when you wish to call a module or chain as a condition’s action.

The settings shown in Figure A check whether a job has finished, and requests the module NOTIFY_SMITH if it has not.

The _first time the condition is true_ is selected because Smith only needs to be notified once.

The REQUEST MODULE action has been expanded to provide the same capabilities as the expanded _awrun_ utility.

The options and prompts for submitting a job are selectable in the Module Details window. To access the Module Details window, click _Details_ as shown in Figure A.
Field Descriptions

The field descriptions are listed below. When ‘Default’ is selected, AppWorx uses the value from the module/chain’s definition.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>The module to request when the action is taken.</td>
</tr>
<tr>
<td>Requestor</td>
<td>The requestor for the module. If no requestor is selected, AppWorx will use the AppWorx login name (e.g. SQLOPER).</td>
</tr>
<tr>
<td>Agent</td>
<td>The agent on which to run the module.</td>
</tr>
<tr>
<td>Alias</td>
<td>Alias for the module. This name appears in the Backlog and History. <strong>Note:</strong> You can use replacement values and substitution variables in this field.</td>
</tr>
<tr>
<td>Priority</td>
<td>Priority of the module relative to other modules in the same queue. If nothing is entered in this field, AppWorx will use the setting defined in the module, or you can specify a value in the range 0-99. Lower numbers are higher priority. <strong>Note:</strong> Zero means no priority, the module will not run.</td>
</tr>
<tr>
<td>DB Login</td>
<td>The login for the module.</td>
</tr>
<tr>
<td>Maximum Run Time</td>
<td>Maximum time the module is allowed to run. If the module runs longer than this amount of time, it will abort.</td>
</tr>
<tr>
<td>Stay in Queue on Abort</td>
<td>Determines whether the module should stay in the Backlog if it aborts.</td>
</tr>
<tr>
<td>Restart on Abort</td>
<td>Determines whether to automatically restart the module once if it initially aborts.</td>
</tr>
<tr>
<td>Use Prompt Defaults</td>
<td>Determines whether the module’s default prompt values will be used. If this option is selected, the Prompts tab will be grayed out.</td>
</tr>
</tbody>
</table>
13.5 Using Conditions to Accomplish Common Tasks

13.5.7 Triggering Two or More Actions on the Same Event

AppWorx can perform only one action for each condition. When you want more than one action triggered by the same event, write multiple conditions with different actions.

You can assign only one action to a condition. However, you can assign many actions to the same event by using multiple conditions. In this example, the first condition will notify Smith the system administrator that the chain is being canceled (see Figure A), and the second condition will cancel the chain (see Figure B).

Both conditions have the **Timing** set to **AFTER**, so AppWorx will evaluate them after the job has executed.

The action frequencies are set to **The first time the condition is true** because the conditions are checked after the module runs.

The order of the two conditions is important because the CANCEL CHAIN action stops condition processing. If the CANCEL CHAIN action comes before the REQUEST MODULE action, the latter module will not be executed and Smith will not be notified.

*Figure A. This condition will notify Smith that the task aborted.*
Figure B. This condition cancels the chain.
13.5 Using Conditions to Accomplish Common Tasks

13.5.8 Running One Module Based on the Failure of Another

You may want to base the running of one module on the unsuccessful completion of another module. You can check for the failure of a job in History with the CHECK HISTORY condition.

You want a certain module, call it module XYZ, to run if and only if another module has failed in the last 30 minutes. The CHECK HISTORY condition examines the History for a job with a specific name and status within an allotted time.

You will need two conditions on module XYZ to implement the requirement correctly: one condition to execute the module if the failed job is found, another to otherwise skip the module.

The Timing is set to BEFORE, so AppWorx will check the conditions before the module is executed.

Both conditions have The first time the condition is true set, because the RUN TASK and SKIP TASK actions only need to be taken one time.

The order of the two conditions is important because the RUN TASK and SKIP TASK actions stop condition processing. If the RUN TASK action comes after the SKIP TASK action, the module will never be executed.

Figure A shows the CHECK HISTORY condition with the RUN TASK action. CHECK HISTORY will check the History for a failed FTP job. If the failed FTP job is found, the condition is true, the module will run and the second condition will NOT be evaluated.

The condition in Figure B will only be evaluated if the first condition is false. If the first condition is false, we want to skip the module. Since 1 = 1 is always true, evaluating the second condition will always skip the module.
**Figure A.** This condition checks the History for a failed FTP job. The module executes if the failed job is found.

**Figure B.** This condition is checked only if the first condition is false. If it is checked, it will skip the module (since 1=1 is always true).
13.5 Using Conditions to Accomplish Common Tasks

13.5.9 Controlling Execution Sequence in a Chain

You can control which modules execute in a chain by using substitution variables in conditions. As an alternative to using conditions, you may wish to add predecessor statements to the first module.

Suppose you want the successful or unsuccessful completion of the first module in a chain to control which of two other modules in the chain executes. For example, if program ‘A’ completes successfully, you want program ‘B’ to run. However, if program ‘A’ fails, you want program ‘C’ to run.

You can accomplish this by creating a substitution variable using the SET SUBVAR action in a condition that is evaluated after the first module completes. The second and third modules use the value of the substitution variable in their conditions with an action of SKIP TASK.

Procedure.

To use a substitution variable to control the execution of chain components:

1. For the first module in the chain, create a pair of conditions to set the value for the substitution variable after the module completes executing. The conditions are shown in Figures A and B.

   The {chain_id} replacement value is used in the substitution variable to make it local to this chain. For more information on localizing substitution

   ![Figure A. Use this condition and the one in Figure B to set the value for the substitution variable.](image)

   ![Figure B. Use this condition and the one in Figure A to set the value for the substitution variable.](image)
variables, see topic 14.6 Creating Local Substitution Variables Using Replacement Values.

2. For the second module in the chain, create the condition shown in Figure C.

![Figure C. If the first module completes successfully, this condition will not be true, the statement will be ignored, and the second module will execute.](image)

If the first module completes successfully, this condition will not be true, the action will be ignored, and the second module will execute.

If the first module does not complete successfully, this condition will be true, the action will be taken—the second module will be skipped.

3. On the second module, include an AFTER condition that deletes the substitution variable (see Figure D).

![Figure D. Delete the substitution variable.](image)
4. For the third module in the chain, create the condition shown in Figure E.
   If the first module completes successfully, this condition will be true, the action will be taken—the third module will be skipped.
   If the first module does not complete successfully, this condition will be false, the action will be ignored, and the third module will execute.

![Figure E](image.png)

*Figure E.* If the first module completes successfully, this condition will not be true, the statement will be ignored, and the third module will execute.

5. On the third module, include an AFTER condition that deletes the substitution variable (see Figure D).

**Another Method**

You could alternately add predecessor statements to the first module to determine which of the next two modules to run based on its successful or unsuccessful completion. You use predecessor statements unless the jobs were running more than once each virtual workday. For an example, see topic 12.7 Example: Using Branching Logic to Determine Which Module Runs.
14
Defining Substitution Variables

14.1 Introduction to Defining Substitution Variables ..................................................... 346
14.2 Defining Static Substitution Variables ................................................................. 348
14.3 Defining Dynamic Substitution Variables .............................................................. 350
14.4 Defining Procedures in Dynamic Substitution Variables ..................................... 352
14.5 Using the Subvar Replacement Values Window ................................................... 354
14.6 Creating Local Substitution Variables Using Replacement Values ....................... 356
14.1 Introduction to Defining Substitution Variables

Substitution variables store values that can be used in modules and chains.

Substitution variables let you store values that can be referenced in modules and chains. The values can be stored in a database table or generated by a SQL statement at the time a job is submitted. The Substitution Variable window is shown in Figure A.

You can use substitution variables to define:
- Prompts in a module or chain
- Conditions for a module or chain
- Output paths in a module definition

The ability to include dynamic substitution variables in conditions lets you control operations based on the state of your corporate database.

Static and Dynamic Substitution Variables

There are two types of substitution variables: static and dynamic. Values for static substitution variables are entered and stored in a database table before the module is executed. Values for dynamic substitution variables are generated by a SQL statement at the time the module is executed. This is the only difference between static and dynamic substitution variables.

Substitution Variables That Ship with AppWorx

AppWorx ships with a number of substitution variables already defined. Several examples are listed below.

- `#current_year`
- `#day_of_week`
- `#first_of_month`
- `#first_of_last`
- `#last_of_last`
- `#last_of_month`
- `#next_monday`
- `#today`
- `#tomorrow`
- `#today`
- `#tomorrow`
Substitution Variables Evaluate as Strings or Numbers

AppWorx evaluates substitution variables as strings or as numbers based on the qualifiers you specify within your conditions. Dates can be evaluated as strings or numbers, but the data type selected in the **Type** field will affect its usage. To ensure proper evaluation of dates as numbers, use Julian dates (dates expressed as the number of days elapsed since January 1, 4713 B.C.) or a YYYYMMDD format. The maximum return size for a substitution variable is 512 characters.

**Using the ‘{ }’ Brackets**

The ‘{ }’ brackets are used to evaluate:

- Environmental variables.
- Replacement values within a substitution variable.

For example: `#job_number={jobid}`

**Warning!** The ‘{ }’ brackets cannot be substituted with ‘[]’ brackets. Substitution variables written with ‘[]’ brackets will not be evaluated.
14.2 Defining Static Substitution Variables

Static variables are stored in a database and retrieved at the time a job is executed. To define a static variable, select New on the Substitution Variables selector window, or set the substitution variable using the SET SUBVAR action in a condition.

Static substitution variables are stored in a database table and retrieved at the time a job is executed. Static variables are generally used for sending information between modules. The information is usually about module status (for example: successful completion, aborted) or information about a module (for example: file name).

Static substitution variables can be defined in the Substitution Variables window or by using the SET SUBVAR action in a condition.

Defining a Static Substitution Variable

To define a static substitution variable:

1. From the Substitution Variables selector window, click New.
   
   AppWorx opens the Substitution Variables window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A.

Note: The SQL Statement box, Check SQL button, and Replacement Values box are not used for static substitution variables. Dynamic substitution variables use SQL statements to return values and perform activities such as retrieving sequences or logging a timestamp reference. For more information, see topic 14.4 Defining Procedures in Dynamic Substitution Variables.
Table A. Field descriptions for the Substitution Variables window

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name must start with a ‘#’ sign, cannot contain blanks, and can be up to 20 characters long.</td>
</tr>
<tr>
<td>Type</td>
<td>The data type, used to validate entries when the substitution variable is used in a prompt as the default, maximum, or minimum value. For more information see topic 11.5 Defining Data Types</td>
</tr>
<tr>
<td>Value</td>
<td>The value for the variable, may be up to 100 characters long. Note: AppWorx does not check the validity of the value you enter.</td>
</tr>
</tbody>
</table>

Defining a Substitution Variable within a Condition

To define a substitution variable with a condition:

1. Choose the SET SUBVAR action as shown in Figure B.
2. Enter the name of the substitution variable, an ‘=’ sign, and the value to be assigned to the variable. The name must start with a ‘#’ sign, cannot contain blanks, and can be up to 20 characters long. Do not put spaces around the ‘=’ sign.

![Figure B. Static substitution variables can be defined in a condition.](image)

The example shown in Figure B will create a substitution variable whose name will include a unique number ‘{chain_id}’ assigned to the chain at run-time. The value of the substitution variable will be ‘GO’.
14.3 Defining Dynamic Substitution Variables

Dynamic Substitution Variables (DSVs) are evaluated at the time a job is executed. A SQL statement generates the values for a DSV. To define a static variable, select New on the Substitution Variables selector window.

Dynamic substitution variables (DSVs) are evaluated at the time a job is executed. A SQL statement generates the value for the DSV. You use DSVs in conditions to evaluate the state of your corporate database and to control execution of a module in a chain. For example, you can check for the number of rows in a table, and delay the module if the count is less than 50. A dynamic substitution variable that returns the date of the first day of the month is shown in Figure A.

Procedure

To define a dynamic substitution variable:

1. From the Substitution Variables selector window, click New.
   
   AppWorx opens the Substitution Variables window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A.
### Table A. Field descriptions for the Substitution Variables window

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name must start with a ‘#’ sign, cannot contain blanks, and can be up to 20 characters long.</td>
</tr>
<tr>
<td>Type</td>
<td>The data type, used to validate entries when the substitution variable is used in a prompt as the default, maximum, or minimum value. For more information, see topic 11.5 Defining Data Types.</td>
</tr>
<tr>
<td>Value</td>
<td>AppWorx ignores this field when you enter the SQL statement.</td>
</tr>
<tr>
<td>Replacement Values</td>
<td>Replacement values retrieve information about AppWorx stored in the AppWorx database. You can add replacement values to your SQL statement by highlighting a value and clicking Insert. For more information on replacement values, see topic 13.4.7 Using Replacement Values in Conditions.</td>
</tr>
<tr>
<td>SQL Statement</td>
<td>The statement should query a database and extract a single (character type) value that will be used for the variable. If the value is a number, you must use the <code>to_char</code> function to convert it to a character. The value returned by the SQL statement must be no longer than 100 characters. The SQL statement cannot be longer than 2000 characters. <strong>Warning!</strong> When the AppWorx master is processing conditions, it stops in the cycle to evaluate each SQL statement. Therefore, it is important that you write concise SQL statements.</td>
</tr>
<tr>
<td>Check SQL</td>
<td>Checks the SQL statement. The value returned by the SQL statement is displayed in the Value field. For more information on modifying data, see topic 14.4 Defining Procedures in Dynamic Substitution Variables.</td>
</tr>
</tbody>
</table>
14.4 Defining Procedures in Dynamic Substitution Variables

Dynamic Substitution Variables (DSVs) can return values and also perform other activities such as retrieving a sequence or logging its own timestamp reference.

Executing database procedures from substitution variables expands the versatility of Dynamic Substitution Variables. A reference to a DSV can return values and also perform other activities such as retrieving a sequence or logging its own timestamp reference.

DSV procedures should never perform lengthy operations as the master process will perform no work during the execution of the procedure. If lengthy procedures are needed, they should be executed through a module.

Sample SQL for Building a Procedure

Create a database procedure and give the AppWorx database login execute authority. A typical Oracle procedure example would be:

```sql
CREATE or REPLACE PROCEDURE test_proc (field1 IN varchar2,
                               field2 IN varchar2,
                               field3 OUT varchar2) IS
BEGIN
  field3 := field1||'='||field2;
END;
```

```sql
grant execute on test_proc to appworx
```

Rules

Keep in mind the following rules when defining a procedure.

- A procedure must not contain more than one output field.
- If the procedure does not contain any output field, the DSV will return a NULL.
- A procedure may contain any number of input fields or none.
- The procedure should not perform lengthy operations.

Required User Option

To use database procedures when defining substitution variables, you must have the `Create Procedure SubVars` user option assigned to you by your AppWorx administrator. For more information on user options, see topic 16.3 Setting User Options.
Defining a Database Procedure in a Dynamic Substitution Variable

To define a procedure in a dynamic substitution variable:

1. Open the Substitution Variables selection window and click Add. AppWorx displays the Add Substitution Variable window shown in Figure A.

2. Enter a name for the variable.
3. Select data type ‘Procedure.’
4. Leave the Value field blank.
5. Enter the procedure call statement, for example:

   \[ \text{prod.test_proc('log.{requestor}', '{jobid}', :result)} \]

   To return a value, the output argument of the procedure must be called :result as shown above. The example procedure uses two replacement values—the Requestor ID and the Job ID. It will return a string that concatenates (combines) the two. The result would return 'log.SQLOPER=4022.00' for a module that was requested by SQLOPER and whose Job ID was 4022.

6. To save the definition and close the window, click OK.
14.5 Using the Subvar Replacement Values Window

To add a substitution variable or replacement variable to a field that includes an ... button, click the ... button next to a field and select a value from the Subvar Replacement Values window.

Substitution variables can be used in a number of places in AppWorx, including prompts and conditions. If a field accepts substitution variables, it will be followed by a ... button.

Procedure

To add a substitution variable or replacement variable to a field that includes an ... button.

1. Click the ... button next to a field, or press F2 when the field is selected.
   
   AppWorx displays the Subvar Replacement Values window shown in Figure A.

2. Select one value from either list.
   
   You can type the first letter of a value to highlight it.

3. Click OK to add the value to a field.

Note: If the cursor is in the middle of a text string when you bring up the Subvar Replacement Values window, AppWorx will place the value at the cursor’s position.

You can enter substitution variables and replacement values in a field by typing the value. Substitution variables are preceded by a '#', and replacement values are written inside curly braces, `{ }`.

Figure A. The Subvar Replacement Values window
14.6 Creating Local Substitution Variables Using Replacement Values

To create a ‘local’ substitution variable for a chain, use the SET SUBVAR action in a condition and include the chain_id replacement value in the substitution variable name.

When creating and assigning values to substitution variables in conditions, there is often a need for the substitution variables to be specific or ‘local’ to a particular run of a chain. This occurs most commonly when a chain is created that uses a substitution variable in its conditions and the chain is then run several times simultaneously. There is the potential for the substitution variable to have its value overwritten and adversely affect all runs of the chain.

Example

SVTEST is a multi-threaded chain that includes three modules: MOD1, MOD2, and MOD3. Each module is single-threaded and has the Ignore Chain option set. MOD1 has two ‘After’ conditions that set substitution variable #flag to ‘SUCCESS’ if MOD1 finishes successfully or ‘FAILURE’ if MOD1 aborts. MOD2 and MOD3 each have a ‘Before’ condition that tests #flag and skips the module if #flag is not set to ‘SUCCESS’. If you run chain SVTEST twice at the same time, the following could happen:

<table>
<thead>
<tr>
<th>Time</th>
<th>SVTEST (Run 1)</th>
<th>SVTEST (Run 2)</th>
<th>#flag Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOD1 starts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>MOD1 starts</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MOD1 finishes</td>
<td>SUCCESS</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>MOD1 aborts</td>
<td>FAILURE</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>MOD2 skipped</td>
<td>FAILURE, so skip</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MOD2 skipped</td>
<td>FAILURE, so skip</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MOD3 skipped</td>
<td>FAILURE, so skip</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>MOD3 skipped</td>
<td>FAILURE, so skip</td>
<td></td>
</tr>
</tbody>
</table>

At Time 1, MOD1-Run1 starts. At Time 2, MOD1-Run2 starts. At Time 3, MOD1-Run1 finishes and its ‘After’ condition sets #flag to ‘SUCCESS’ because the module finished successfully. At Time 4, MOD1-Run2 aborts and its ‘After’ condition sets #flag to ‘FAILURE’ because the module aborted. At Times 5-8, the ‘Before’ conditions on the remaining modules all force a skip because #flag is set to ‘FAILURE.’ This is not the desired result - MOD2 and MOD3 for Run 1 should have run because MOD1-Run1 finished.
Solution

To avoid this problem, you need a substitution variable that is specific or ‘local’ to each run of SVTEST. You can create such a substitution variable using the chain_id AppWorx replacement value.

For each run of a chain, the chain_id replacement value is a unique number. You can use this unique number as part of the substitution variable name. It also helps to remember that AppWorx will automatically create substitution variables used in conditions if they don’t already exist.

Below are several examples of conditions and how they can be improved with the chain_id replacement value:

<table>
<thead>
<tr>
<th>Without {chain_id}</th>
<th>With {chain_id}</th>
</tr>
</thead>
<tbody>
<tr>
<td>After:STATUS = FINISHED SET SUBVAR #flag = SUCCESS</td>
<td>After:STATUS = FINISHED SET SUBVAR #flag{chain_id} = SUCCESS</td>
</tr>
<tr>
<td>Before:#result != SUCCESS SKIP TASK</td>
<td>Before:#result{chain_id} != SUCCESS SKIP TASK</td>
</tr>
<tr>
<td>After:1 = 1 DELETE SUBVAR #value</td>
<td>After:1 = 1 DELETE SUBVAR #value{chain_id}</td>
</tr>
</tbody>
</table>

In each case, the substitution variable (#flag, #result, #value) is augmented with the unique chain_id number as part of its name. #flag might become #flag69, #result might become #result888, and #value might become #value21107.

Making the substitution variable name unique for each run of the chain eliminates any chance that the substitution variable will be overwritten when a chain runs concurrently with itself. There is one caveat about using {chain_id} to uniquely name substitution variables. You must be sure to delete the unique substitution variable when the chain ends, otherwise you can end up with a large number of substitution variables. For example, if a chain that uses #flag{chain_id} in its conditions runs once an hour every day, at the end of a week there would be 168 new substitution variables without a use. To delete a substitution variable, add a DELETE SUBVAR condition at the end of the chain at a position where the condition is guaranteed to be evaluated. The condition must always evaluate true to guarantee the DELETE SUBVAR action is taken. The table above contains an example of such a condition at the bottom of the With {chain_id} column.

'1 = 1' always evaluates true and the DELETE SUBVAR action acts upon #value{chain_id}.
Chapter 14: Defining Substitution Variables
Section IV

Administration

Chapter 15: Defining Output Devices ................................................................. 361
Chapter 16: AppWorx Security ...................................................................... 375
Chapter 17: Working with Agents ................................................................. 403
Chapter 18: Administering Queues ............................................................. 421
Chapter 19: Exporting and Importing Objects ........................................... 429
15

Defining Output Devices

15.1 Introduction to Defining Output Devices ............................................................... 362
15.2 Defining Single Printers ........................................................................................ 364
15.3 Defining Distribution Lists ..................................................................................... 366
15.4 Defining Printer Groups ........................................................................................ 368
15.5 Defining Print Spoolers ......................................................................................... 370
15.6 Adding Print Options to Spooler Definitions ......................................................... 372
15.1 Introduction to Defining Output Devices

In AppWorx, a printer is any output device. A printer can represent a single printer, or through the use of distribution lists, multiple printers. You grant access to a printer by assigning to it one or more user roles. You can assign one or more printer definitions to a printer group.

In AppWorx, printers are assigned to modules and chains. AppWorx printers can be any output device including printers, faxes, and email. A printer definition can represent a single device or a group of devices. Access to a printer is granted by assigning roles to the printer.

Single Printers and Distribution Lists

A printer definition can define a single device or a group of devices using a distribution list. In Figure A, ALL_LASER represents a distribution list. If a distribution list is assigned to a module or chain, output is sent to each output device in the list.

Printer Groups

Printer groups are used to define organizational classes of output devices. When you define a printer, you assign the printer to one or more printer groups. When you assign a printer group to a module, output from the module can be sent only to the devices included in the printer group.

Spoolers

In AppWorx, a spooler is the interface to an output device. Associated with the spooler is a spooler script. The script builds the correctly formatted command that actually sends the output to the output device. A spooler can define any device that processes output such as a printer, fax, and email. A spooler can be assigned to one or more printers.

Troubleshooting Printer Problems

For a table of symptoms, causes and actions for printing troubles, see Appendix C: Troubleshooting Printer Problems.
15.2 Defining Single Printers

To add a new printer definition, you must enter a name and description for the printer, select a spooler, and assign the printer to one or more printer groups. If you have not already defined a printer group and a spooler, you can do so as you create the printer definition.

To add a new printer definition, you must:

- Enter name and description.
- Select spooler (and printer text option if defined).
- Select an auto print option.
- Assign one or more printer groups.

Note: AppWorx roles control access to printers. If you do not have access to them, see your AppWorx administrator.

If you want the printer definition to define multiple printers, you can create a distribution list.
Instructions for creating a distribution list are described in topic 15.3 Defining Distribution Lists.

Procedure

To add a new printer definition:

1. From the Printers selector window, click New.
   AppWorx Displays the Select printer type window shown in Figure B.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Select the Single printer option.
   AppWorx displays the Printers window shown in Figure A.

3. Complete the fields using the information in Table A.
**Table A. Field descriptions for defining a single printer on the Printers window**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the output device.</td>
</tr>
<tr>
<td>Description</td>
<td>Description for the output device.</td>
</tr>
<tr>
<td>Spooler</td>
<td>The spooler script that actually executes the print command and passes user defined parameters to the printer, email, fax, etc. The ‘lp’ spooler is assigned to the printer in Figure A. The spooler defines the names of the static options fields, as well as basic command design. For information on defining spoolers in AppWorx, see topic 15.5 Defining Print Spoolers.</td>
</tr>
<tr>
<td>Printer option</td>
<td>Allows you to request a specific variable, setting, address, or orientation. Determined by the print options defined for the spooler. For information on defining print options, see topic 15.6 Adding Print Options to Spooler Definitions.</td>
</tr>
<tr>
<td>Static options</td>
<td>The label fields defined for the spooler, which allow you to define parameters for a module. In Figure A, the <strong>Dest</strong>, <strong>Form</strong>, and <strong>Queue</strong> options are defined. For information on defining static options, see topic 15.5 Defining Print Spoolers.</td>
</tr>
<tr>
<td>Command</td>
<td>This field is read only. As you fill in the static options, it displays the actual line executed by the agent.</td>
</tr>
</tbody>
</table>
| Auto print options| • Active: The output is sent to the printer.  
• Inactive: The output will not be sent to the printer. The output will, however, be sent to the log file for viewing.  
• Alternate printer: When you select an alternate printer, it will be used instead of the printer you are defining. This feature is useful if you need to temporarily take the printer offline, maybe for maintenance, and want to redirect output to another printer. |
| Unassigned groups| Use the arrow keys to assign/unassign the printer to one or more printer groups. For information on defining printer groups, see topic 15.4 Defining Printer Groups.                                                                                      |
| Assigned groups  |                                                                                                                                                                                                                                                                       |
15.3 Defining Distribution Lists

A printer definition can represent a single device or multiple devices. To specify multiple devices, create a distribution list and assign two or more printers to it.

If you want a printer definition to represent multiple printer devices, you must create a distribution list for the definition. A distribution list contains the names of other printer definitions.

For example, if a distribution list is selected for a module, that module’s output will be printed to all printers listed in the distribution list, even if the module’s requestor does not normally have access to one or more of the printers. If you create a distribution list for a printer definition, you cannot assign a print spooler to the definition. Instead, each printer added to the distribution list uses its own spooler definition. This allows the distribution list to include a variety of different printer devices. If a distribution list is selected in the Submit window, the function selected is overridden by the contents of the distribution list, since the distribution list may contain printers with different output function settings.

Procedure

To create a distribution list object:

1. From the Printers selector window, click **New**.
   
   AppWorx Displays the Select printer type window shown in Figure B.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Select the **Distribution List** option.
   
   AppWorx displays the Printers window shown in Figure A.

3. Enter a name and description for the distribution list.

4. Assign/Unassign the distribution list to one or more printer groups, using the arrow keys. For information on defining printer groups, see topic 15.4 Defining Printer Groups.
### Adding Printers to a Distribution List

To add a printer to a distribution list:

1. Select a printer and its data using the fields described in the table below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printer</td>
<td>The printer to be added. The drop-down list includes all printers you have previously defined.</td>
</tr>
<tr>
<td>User</td>
<td>Determines who can view the output, and will be displayed in the header of the printout.</td>
</tr>
<tr>
<td>Function</td>
<td>Determines how output is handled. On the Output window, output with a status other than LOG must be queried for before it can be viewed. All output can be viewed from the Jobs/Explorer window. LOG: Output is available for viewing from the Jobs/Explorer and the Output windows with an ‘N’ status. Use this option for end users who want to see their output immediately online. PRINT: Output is printed. The output is available for viewing from the Jobs/Explorer and the Output windows with a ‘P’ status. STORE: Output is stored and can be viewed from the Jobs/Explorer and the Output windows with a ‘D’ status. Use this option for output that may be needed for historical reference or troubleshooting.</td>
</tr>
<tr>
<td>Printer option</td>
<td>Allows you to request a specific variable, setting, address, or orientation. Determined by the print options defined for the spooler. For information on defining print options, see topic 15.6 Adding Print Options to Spooler Definitions.</td>
</tr>
<tr>
<td>Copies</td>
<td>Number of copies.</td>
</tr>
</tbody>
</table>

When you enter or change a field value a red triangle is displayed on the General tab to signify unsaved changes. For more information, see topic 8.3 Updating Unsaved Changes.

2. Click **Add** to include the printer to the distribution list.

### Editing, Deleting, and Clearing a Distribution List

To edit or delete an entry in a distribution list, highlight the entry and click the appropriate button on the right. To delete all entries, click **Clear**.
15.4 Defining Printer Groups

Printer groups are used to define organizational classes of output devices. When you define a printer, you assign it to one or more printer groups. Output from a module can be sent only to the devices included in the selected printer group.

Printer groups are used to define organizational classes of output devices. Output devices can include, but are not limited to, printers, fax machines, and email. Printer groups are used in printer definitions and module definitions. When you assign a printer group to a module, output from the module can be sent only to the devices included in the printer group.

Note: AppWorx roles control access to printer groups. If you do not have access to them, see your AppWorx administrator.

Printer groups are different from printer definitions. Printer groups are used to group similar printer definitions together into useful sets—they are basically a classification mechanism. Printer definitions contain the specific output device attributes for a single output device. A distribution list is a special case printer definition that contains the output device attributes for several output devices.

AppWorx ships with the ANY and SYSOUT printer groups already defined. AppWorx will use the ANY printer group as a default if no other printer group is defined. You will probably want to define additional printer groups based on security issues (for example: a payroll check printer), printer capabilities (for example: line, dot matrix, laser), location (for example: first floor, accounting) or some other useful classification.
**Procedure**

To add a new printer group:

1. From the Printer Groups selector window, click **New**.
   
   AppWorx opens the Printer Groups window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 *Adding, Editing, and Deleting AppWorx Objects*.

2. Enter a name and description for the printer group.
   
   The name can be up to 25 characters long. The description can be up to 30 characters long.

3. To accept the information for the new printer group, click **OK**.

**Assigning Printers to a Printer Group**

You assign single printers and distribution lists to a printer group using the Printers window (see Figure B). When a printer is assigned to a printer group, it is included the **Printers** box for that printer group. There are four printers included in the printer group shown in Figure A.

![Figure B. You assign printers to a printer group from the Printers window.](image)
15.5 Defining Print Spoolers

A print spooler builds the print command that is sent to an output device. Associated with the spooler is a spooler interface script. The interface script builds the correctly formatted command that actually sends the output to the output device.

In AppWorx, a spooler is an interface between AppWorx and an output device. Associated with the spooler is a spooler interface script. The interface script builds the correctly formatted command that actually sends the output to the output device. A spooler can define any device that processes output such as a printer, fax, and email. A spooler can be assigned to one or more printers.

AppWorx comes with an *lp* spooler defined for the standard UNIX *lp* command and the Windows *print* command. An LPP interface script supports the *lp* spooler. The script is located in the `$AW_HOME/exec` subdirectory. The spooler must pass the correct information to the spooler interface script to enable it to run the appropriate command.

**Note:** AppWorx roles control access to spoolers. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To add a new printer spooler:

1. From the Spoolers selector window, click **New**.
   AppWorx opens the Spoolers window shown in Figure A.
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.
2. Complete the fields using the information in Table A.
### Table A. Field descriptions for the Spoolers window

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name for the spooler interface command. We recommend using the name of the printing command as the name of the spooler, for example <code>lp</code> for the <code>lp</code> UNIX command.</td>
</tr>
<tr>
<td>Command</td>
<td>The spooler command that will be sent as the first element in the interface command. Each spooler command calls a shell script in the <code>$AW_HOME/exec</code> subdirectory. If the shell script does not exist, you must create it. The LPP shell script for the default <code>lp</code> spooler is shown below.</td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>file=$1</td>
</tr>
<tr>
<td></td>
<td>shift</td>
</tr>
<tr>
<td></td>
<td>eval &quot;lp $* $file&quot;</td>
</tr>
<tr>
<td></td>
<td>This shell script takes the information passed to it by AppWorx and builds an <code>lp</code> command with correct syntax (in this case, it reads the file name as the first argument and moves it to the end of the UNIX <code>lp</code> command).</td>
</tr>
<tr>
<td></td>
<td>Note: Be sure to make your scripts executable.</td>
</tr>
<tr>
<td>Title</td>
<td>Used to add a title to the banner page of the printout, enter the appropriate character sequence here. AppWorx will substitute the AppWorx requestor name for the variable “%USR%”.</td>
</tr>
<tr>
<td>Space character</td>
<td>The character you would like to use as a placeholder for the space character. Instead of using the spacebar on your computer, type the character in this field. The default is the '#' character.</td>
</tr>
<tr>
<td>Copies</td>
<td>Specifies the number of copies to print when you submit a module with the Submit window, enter the appropriate prep value character sequence in the Copies Prep field. For UNIX <code>lp</code>, this is ‘-n’. If you do not enter a flag in this field, any number entered into the Copies field in the Submit window is ignored.</td>
</tr>
<tr>
<td>Static options</td>
<td>Fields 1-3 are used to define parameters for a module. These static option labels are displayed on the Printer window. In Figure A, the Dest, Form, and Queue options are entered. The fields to the right of the 1-3 fields allow you to specify the format of each static option. If you leave these fields blank, they will not populate their respective fields on the Printers window and AppWorx will not insert their flag value to their modules.</td>
</tr>
<tr>
<td>Print option box</td>
<td>Allows you to request a specific variable, setting, address, or orientation. Determined by the print options defined for the spooler. For information on defining print options, see topic 15.6 Adding Print Options to Spooler Definitions.</td>
</tr>
<tr>
<td>Command line</td>
<td>This field is read only. As you fill in the fields, it displays the actual line executed by the agent.</td>
</tr>
</tbody>
</table>
15.6 Adding Print Options to Spooler Definitions

Print options allow users to request a specific variable, setting, address, or orientation during run-time. They are defined on the Spooler window.

In AppWorx, a spooler is an interface between AppWorx and an output device. AppWorx allows for static and print options to be defined for spoolers. They are set by the user at the time they select the printer.

You create printer options on the Spooler window. Using this option, users can select a value from a predefined list or enter a value when defining/requesting modules and chains, or when adding a user. This feature is handy when creating a single printer definition that you want to use with multiple settings (for example: portrait and landscape printer settings, or email addresses). The printer option, if used, is always the second argument passed on the spooler command line (after <file>). Figure A illustrates the four versions of the print option described in Table A.

Figure A. You can choose from four Print option selections in the Spooler window. Clockwise from top right; Fill in, Fill in List, List and None.
Table A. The Print option fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection type</td>
<td><strong>None:</strong> Variable print options will not be used.</td>
</tr>
<tr>
<td></td>
<td><strong>Fill-in:</strong> Users enter their own value.</td>
</tr>
<tr>
<td></td>
<td>Be sure you design your spooler command script to anticipate the use (or omission) of the variable printer option.</td>
</tr>
<tr>
<td></td>
<td><strong>List:</strong> Users can select a value from a list of choices.</td>
</tr>
<tr>
<td></td>
<td><strong>Fill-in/List:</strong> Users can select from a list of choices or enter their own value.</td>
</tr>
<tr>
<td>Option</td>
<td>Insert individual values into this field. Click <strong>Add</strong> to add the value to the option list.</td>
</tr>
<tr>
<td>Option list</td>
<td>Shows listing of default values.</td>
</tr>
</tbody>
</table>

In the Printer window shown in Figure B, AppWorx builds the spooler command as you fill in the static and variable options with values. As you make a change, AppWorx reflects that change in the **Command** field. Figure B shows an e-mail print device.

![Figure B. An e-mail print device.](image)

**Note:** The **Printer option** field only appears in other AppWorx windows if a print option has been defined for the spooler assigned to the printer.
16

AppWorx Security

16.1 Introduction to AppWorx Security ................................................................. 376
16.2 Defining Users ............................................................................................. 378
16.3 Setting User Options .................................................................................... 380
16.4 Working with Roles ...................................................................................... 382
  16.4.1 Defining Roles ....................................................................................... 384
  16.4.2 Assigning Objects to a Role .................................................................... 386
  16.4.3 Understanding Role Authorities ............................................................ 388
  16.4.4 Assigning Roles to an Object ................................................................. 390
16.5 Sample Roles ............................................................................................... 392
  16.5.1 Programmer Roles .................................................................................. 394
  16.5.2 Operator Roles ....................................................................................... 396
  16.5.3 End User Roles ...................................................................................... 398
16.6 Defining Logins ............................................................................................ 400
16.1 Introduction to AppWorx Security

Using the AppWorx security features, you can control access to AppWorx, objects within AppWorx, and to databases accessed by modules.

Security is always a major concern with client/server programs. AppWorx lets you control security on three levels:

- Access to AppWorx is controlled with user logins.
- Access to AppWorx functions is controlled with roles.
- Access to databases and hosts is controlled with logins.

Using the three security levels, you can fine tune AppWorx to give different groups access to only the areas of AppWorx they need. The different access controls are illustrated in Figure A.

![Figure A. AppWorx lets you control security on three levels](image)

Defining Users

When you add a user to AppWorx, you define the user’s name, password, login group, maintenance role, and default output directory. You can assign AppWorx users specific options such as View all User Outputs for the Output window. For more information on users, see topic 16.2 Defining Users.
Defining Roles

You control access to AppWorx windows and objects using roles. Think of roles as containers to which you add objects and users. Users have access to all objects with which they share a container. Users and other AppWorx objects can be placed in more than one container. You can further restrict access by designating Edit authorization to some roles and not to others. The number of objects and users you can assign to a role is unlimited. Objects and users can be assigned to more than one role. Users have access to all the objects in all the roles to which they are assigned. For more information on roles, see topic 16.4.1 Defining Roles.

Defining Logins

Use an AppWorx login to protect the actual database or host login and password while giving users access to it. If a program run by a module requires access to a database or host, you must create an AppWorx login for it. The login includes the login name, password, type of database, database ID or network connect string, and an encryption flag. The encryption flag is set by the system when you update or add a login. After you create a login, other users can assign the login to modules and chain components. For more information on logins, see topic 16.6 Defining Logins.
16.2 Defining Users

Access to AppWorx is controlled with user logins.

**Note:** AppWorx roles control access to the Users window. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To add a new user:

1. From the Users selector window, click **New**.
   
   AppWorx opens the Users window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A.

**Table A. User General tab field descriptions**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>The name the user will enter on the AppWorx Login Window. Entry can be 30 characters long.</td>
</tr>
<tr>
<td>First Name</td>
<td>The user’s first and last name or description.</td>
</tr>
<tr>
<td>Last Name/Description</td>
<td></td>
</tr>
<tr>
<td>Active</td>
<td>Designates the user as active or inactivate.</td>
</tr>
<tr>
<td>Password</td>
<td>The password for the user. Entry can be 30 characters long.</td>
</tr>
<tr>
<td>Expire interval</td>
<td>The number of days until the user’s password will expire (forcing them to specify a new one).</td>
</tr>
<tr>
<td>Last modified</td>
<td>This non-edit field displays the date and time the password was last changed.</td>
</tr>
</tbody>
</table>
Now that you have added a user object to AppWorx, the next step is to assign user options to it. For more information, see topic 16.3 Setting User Options.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| Awexe range      | Numbers that correlate with a range of options listed in the awserver_sql.dat file located in the data folder. Users will have access to the functions assigned to them while in a UNIX session. Ranges are listed below:  
• 1000-1999: Used for AppWorx background processes.  
• 2000-2999: Utilities for the general user.  
• 3000-4999: Reserved for consulting.  
• 5000-9999: Used for custom definitions.  
If you enter 2000-2099, the user will have access to all the AppWorx Command Line Interface (CLI) functions. However, to change any data, they will be required to supply an AppWorx user name and password. For more information on the CLI, see chapter 7: *Using the AppWorx Command Line Interface*.  
**Note:** Users assigned to a remote agent require an awexe range for the agent to start.                                                                                                                                                                                                 |
| Default login    | AppWorx uses this field to specify the login for the user when the user is identified as the requestor of a module that requires this login.                                                                                                                                                                                                     |
| Maintenance role | This option lets you specify a role that will automatically be assigned to all objects created by the user. To select a role from the drop-down list, it must be assigned to the user. For more information, see topic 16.4.2 Assigning Objects to a Role.  
With the objects and the user assigned to the same role, the user can edit the objects he creates. If you do not assign a Maintenance Role to a user, and the user creates a module or chain, he will not be able to edit the object after he creates it.  
Whenever a user creates a new object, the object is automatically assigned to the DBA role even if the user does not have DBA authority. This ensures that the AppWorx administrator can get to the object to edit or delete it.  
You can assign the user to one or more roles.                                                                                                                                                                                                                               |
| Printer          | Specifies a default printer for jobs run under a user’s name when a printer group is assigned to a module or chain component.                                                                                                                                                                                                              |
| Print option     | Allows users to request a specific variable, setting, address, or orientation. This field appears only when a variable print option has been defined for the selected printer.                                                                                                                                                                                                                           |
| Output Directory | You can specify a default directory for job output (but not system output) run under a user’s name. However, AppWorx must have access to the directory. If you specify a directory, this setting overrides the Output directory specified for a module. Usually this field is left blank.                                                                                              |
16.3 Setting User Options

User options control user access to AppWorx features such as the Hide feature in the Output window.

The AppWorx user options control user access to specific features. The user options are listed in Table A. By default no user options are assigned to a new user.

![Figure A. A User with five user options assigned.](image)

When you initially install AppWorx, the script automatically creates the following AppWorx users.

- SQLOPER (the default user), AppWorx automatically assigns several user options to this login.
- A user named after the login you are using to install AppWorx. This user will include have no default user options.

**Procedure**

To assign options to a user:

1. From the Options tab of the Users window, select the values you want to use.
   
   For information on assigning options, see topic 2.2 *Working in the AppWorx Windows*.

2. To accept the values, click OK.
### Table A. User options

<table>
<thead>
<tr>
<th>User Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Chain Requestors</td>
<td>When assigned, all currently defined users will be displayed in the Requestor list box on the Schedule tab for modules and chains. If not assigned, only the current user will be displayed. For more information on schedules, see topic 10.2 Entering General Information for Schedules.</td>
</tr>
<tr>
<td>All User Backlog Edits</td>
<td>Allows the user to edit job parameters for all jobs in the Backlog. For information on editing job parameters, see topic 5.9 Viewing and Editing Job Details.</td>
</tr>
<tr>
<td>Create Procedure SubVars</td>
<td>Allows the user to use database procedures when defining substitution variables. For information on defining procedures, see topic 14.4 Defining Procedures in Dynamic Substitution Variables.</td>
</tr>
<tr>
<td>Deny FTP Option</td>
<td>When assigned the user will not have access to the File Transfer option when printing from the File Viewer. For more information on the FTP option, see topic 4.5 Printing Job Output.</td>
</tr>
<tr>
<td>Deny Outmgr Hide</td>
<td>When assigned, the user will not have access to the Hide button on the Output window—the user will not be able to hide output. For more information on the Output window, see topic 4.2 Working with the Output Window.</td>
</tr>
<tr>
<td>Hide RO Prompts</td>
<td>When assigned, the user will not be shown unchangeable prompts on the Submit window, the Jobs/Explorer window, or when defining modules and chains. Prompts without Allow Changes set will be read only.</td>
</tr>
<tr>
<td>Manage all Agents</td>
<td>Allows the user to start and/or stop all agents from the Jobs/Explorer window. The user will also need appropriate role access to the agents. For more information on starting and stopping agents, see topic 5.12 Managing Agents.</td>
</tr>
<tr>
<td>Request Queues</td>
<td>The user can select a different queue when submitting jobs on an ad hoc basis. For more information on submitting jobs, see topic 3.3 Editing and Submitting Modules and Chains.</td>
</tr>
<tr>
<td>View all User Outputs</td>
<td>Output from all users will be listed in the Output window for this user. For more information on the Output window, see topic 4.2 Working with the Output Window.</td>
</tr>
</tbody>
</table>
16.4 Working with Roles

You control access to AppWorx windows and objects using roles. Think of roles as containers to which you add objects and users. Users have access to all objects with which they share a container. Users and other AppWorx objects can be placed in more than one container. You can further restrict access by designating Edit authorization to some roles and not to others.

Organizing Roles

Roles give you unlimited flexibility in assigning access to AppWorx objects. However, you may want to start with some standard roles until you have worked with AppWorx and have had time to think through a more elaborate set of roles. The top row in Figure A illustrates three typical roles: programmer, operator, and end user.

![Roles Diagram]

*Figure A. Roles can be thought of as containers. After creating a role, you add objects and users to the role. You might also customize your roles into edit and non-edit.*

Figure A also shows two additional programmer roles: edit and non-edit. The two roles make it possible to give programmers read only access to some objects, and edit access to other objects.
The programmers would be assigned to both roles. For example, you might give programmers read only access to:

- Objects that ship with the product (for example: system modules and chains).
- Objects that might be created by an AppWorx administrator (for example: printers).
- Objects that might be created by a database administrator (for example: logins).
- Certain objects such as queues (to give access the Jobs/Explorer window).

On the other hand, you would give programmers edit access to the modules and chains they create.

By having the two roles, you have the flexibility to give a group of users the access they require to accomplish their job. You would most likely want an edit and non-edit role for operators as well. End users may only require a single non-edit role because they would not be creating objects.

**DBA Role**

The Database Administrator (DBA) role is defined when you install AppWorx. Usually the AppWorx administrator is assigned to the DBA role. It has access to all AppWorx objects. You cannot edit or delete this role. Whenever a user creates an object, it is automatically assigned to the DBA role. The only objects you can assign/unassign to this role are users. Users with the DBA role may have different user options assigned to them.

**Subtopics**

The following topics explain how to define roles and add objects to them.

- 16.4.1 Defining Roles
- 16.4.2 Assigning Objects to a Role
- 16.4.3 Understanding Role Authorities
- 16.4.4 Assigning Roles to an Object
16.4 Working with Roles

16.4.1 Defining Roles

Creating a role is a two step process. First you define the role, then you assign objects to it. You can create as many roles as you need. A role can give users read-only access to objects, or read and edit access using the Edit option.

After planning the roles you want to create, you are ready to define the roles. You can create as many roles as you need. Roles can incorporate any combination of objects, including other roles. The Roles window is shown in Figure A.

![Figure A. The Roles window.](image)

Procedure

To add a role:

1. From the Roles selector window, click **New**.
   
   AppWorx opens the Roles window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A.
Table A. The Roles window fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the role</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the role</td>
</tr>
</tbody>
</table>
| Edit  | When checked, this box allows the users assigned to the role to do two things:  
• Edit and delete the objects assigned to it.  
• Add objects if the corresponding role authorities are assigned to the user.  
For more information on role authorities, see topic 16.4.3 Understanding Role Authorities. |

Next Step

After defining a role, the next step is to add objects to it. For more information, see topic 16.4.2 Assigning Objects to a Role.
16.4 Working with Roles

16.4.2 Assigning Objects to a Role

To assign objects to a role, select the role and click the Assign tab.

After creating a role, you assign objects to it by clicking the Assign tab. You can assign any type and number of objects to a role. However, it is usually a good idea to organize roles so each defines a group of similar objects. For example, you might want to create one role to define a set of modules, a second role to define a set of printers, and a third role to define a set of role authorities.

Procedure

To assign objects to a role:

1. From the Roles window, select the Assign tab (see Figure A).
2. Select an object type from the Object types list.
   Selecting ‘All’ will display all roles.
   Note: Role authorities control user access to the AppWorx windows. For more information, see topic 16.4.3 Understanding Role Authorities.
3. Select the objects you want to include in the role.
   For information on assigning options, see topic 2.2 Working in the AppWorx Windows.
Setting the Module/Chain Option

The **Both**, **Request**, and **Output** radio buttons on the bottom of the screen are only used when assigning modules and chains to a role. When you add them to a role, you can choose one of the options described in the table below.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both</td>
<td>Users can submit these modules and chains and view the output with the file viewer.</td>
</tr>
<tr>
<td>Request</td>
<td>Users can submit these modules and chains, but not view their output.</td>
</tr>
<tr>
<td>Output</td>
<td>Users can view the output of these modules and chains, but not request them.</td>
</tr>
</tbody>
</table>
16.4 Working with Roles

16.4.3 Understanding Role Authorities

Role authorities control user access to the AppWorx windows. You cannot add, edit, or delete them. But you can assign them to your roles. Whether role authorities for AppWorx objects are assigned to a role with Edit authorization determines whether the users who are also assigned to that role can add those objects to AppWorx.

Role authorities are a special object type that control user access to the AppWorx windows. You cannot add, edit, or delete role authorities like other AppWorx objects. But you can assign them to roles (see Figure A).

**Edit Authorization is Required to Add Objects**

Whether role authorities for AppWorx objects are assigned to a role with Edit authorization determines whether the users who are also assigned to that role can add those objects to AppWorx (see Figure B).

Edit Authorization also allows users to edit and delete any objects assigned to the role. For more information, see topic 16.4.1 Defining Roles.
Using Edit Authorization and Role Authority with Roles

The table below shows several role assignments for users of modules that include different *Edit* authorizations and role authorities. If you were creating these roles, you would include other objects and role authorities.

<table>
<thead>
<tr>
<th>If you want users to be able to:</th>
<th>Assign them to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Add modules to AppWorx.</td>
<td>One or more roles with <em>Edit</em> authorization including:</td>
</tr>
<tr>
<td>• Edit and delete modules.</td>
<td>• The Modules role authority.</td>
</tr>
<tr>
<td>• Assign modules to chains.</td>
<td>• The modules they can edit and delete.</td>
</tr>
<tr>
<td></td>
<td>If necessary, one or more roles without <em>Edit</em> authorization including:</td>
</tr>
<tr>
<td></td>
<td>• Modules that these users can add to a chain, but not edit or delete.</td>
</tr>
<tr>
<td>• Edit and delete modules.</td>
<td>One or more roles with <em>Edit</em> authorization including:</td>
</tr>
<tr>
<td>• Assign modules to chains.</td>
<td>• The modules they can edit and delete.</td>
</tr>
<tr>
<td></td>
<td>One or more roles without <em>Edit</em> authorization including:</td>
</tr>
<tr>
<td></td>
<td>• The Modules role authority.</td>
</tr>
<tr>
<td></td>
<td>• Modules that these users can add to a chain, but not edit or delete.</td>
</tr>
<tr>
<td>• View module definitions.</td>
<td>One or more roles without <em>Edit</em> authorization including:</td>
</tr>
<tr>
<td>• Assign modules to chains.</td>
<td>• The Modules role authority.</td>
</tr>
<tr>
<td></td>
<td>• Modules that these users can add to a chain, but not edit or delete.</td>
</tr>
<tr>
<td>• Assign modules to chains.</td>
<td>One or more roles without <em>Edit</em> authorization including:</td>
</tr>
<tr>
<td></td>
<td>• Modules that these users can add to a chain, but not edit or delete.</td>
</tr>
</tbody>
</table>
16.4 Working with Roles

16.4.4 Assigning Roles to an Object

With the Roles tab, you can assign any AppWorx object to a role.

Topic 16.4.2 Assigning Objects to a Role discussed assigning objects to a role. But roles can also be assigned to an object. For example, you can assign a role to a chain from the Chains window. Because AppWorx is object-oriented, there is no difference between assigning a role to an object, or an object to a role.

Procedure

To assign roles to an object:

1. From any object’s definition window select the Roles tab
   In Figure A the Roles tab is selected for a queue.
2. Select the roles you want to assign to the object.
   For information on assigning options, see topic 2.2 Working in the AppWorx Windows.
16.5 Sample Roles

Sample roles are described for programmers, operators, and end users.

In this topic, sample roles are described for programmers, operators, and end users. They are provided to help you begin thinking about how you want to set up roles for your system. Feel free to modify these samples to meet your needs.

General Guidelines

Below are some general guidelines that apply to roles.

- For each class of user (programmers, operators, etc.), set-up two roles: one with and one without Edit authorization.
- Assign objects you want a user to edit to the role with Edit authorization.
- Assign objects you want a user to view, but not edit, to the role without Edit authorization.
- For each user, assign the role with Edit authorization as their Maintenance role. This ensures the user will have access to the objects they create.

Two Roles are Better Than One

In general, setting up an edit role and a non-edit role for each type of user will give you adequate flexibility in controlling a user’s access to objects.

The role with Edit authorization would include objects you want the user to be able to edit. For example, you might include the Modules role authority and perhaps several modules that programmers use. This would allow the programmers to edit some existing modules, and create new ones. The role with Edit authorization would also be assigned as the Maintenance role for the user. This would let the programmers edit any of the modules they create.

The role without Edit authorization would include objects you want the user to view, but not edit. Using the programmer role as an example, you would include the modules you want the programmer to view.

Subtopics

The following subtopics outline typical roles for three user groups:

16.5.1 Programmer Roles
16.5.2 Operator Roles
16.5.3 End User Roles
16.5 Sample Roles

16.5.1 Programmer Roles

For programmers, create an edit role that contains the role authorities, and a non-edit role for all other objects the programmers need to access but not edit.

A programmer’s key responsibility is creating modules and chains. To create modules and chains, programmers need to create other objects such as substitution variables, program types, and data types. Programmers must also be able to select, but not necessarily create, printers, logins, and queues. You can give programmers the access they need using two roles: one edit, the other non-edit (read only).

Typically, a programmer should be able to create and edit the following objects:

- Modules
- Libraries
- Substitution Variables
- Chains
- Program Types
- Applications
- Data types
- Applications
- Data types
- Calendars
- Agent Groups
- Printer Groups
- Logins
- Queues
- Agents
- Printers
- Submit modules and chains.
- Monitor the progress of jobs in the Backlog and History.
- View output on the Output window.

Creating the Programmer Edit Role

You should assign the following role authorities to the role with Edit authorization. This role could be named Programmer_Edit.

- Modules
- Libraries
- Substitution Variables
- Outputs
- Chains
- Program Types
- Requests
- Explorer
- Applications
- Data types
- Explorer

The programmers will also need access to the objects they create including modules, chains, substitution variables, data types, and calendars. However, access to these objects will be taken care of by assigning this role as the Maintenance role for each programmer. When a programmer creates one of these objects, it will automatically be assigned to the Maintenance role.
Creating a Programmer No Edit Role

The table below shows the objects that should be assigned to the role without Edit authorization. This role could be named programmer_no_edit. Programers will need to select these objects when creating modules and chains, but there is no need for the programmers to edit them. For example, a programmer will need to assign an agent to a module to complete the module definition, but the programmer does not have to create or edit agents. Creating and editing agents will probably be done by the AppWorx administrator.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendars</td>
<td>Calendars created by the AppWorx administrator and operations.</td>
</tr>
<tr>
<td>Chains</td>
<td>Chains that ship with the product, for example SYSTEM and PRODSCH.</td>
</tr>
<tr>
<td>Data Types</td>
<td>Data types that ship with the product, for example Character, Numbers, Dates, and Modules.</td>
</tr>
<tr>
<td>Logins</td>
<td>All logins</td>
</tr>
<tr>
<td>Modules</td>
<td>Modules that ship with the product, for example FTP, LOADER, PRODSCH, SODELETE, and HISTORY_PURGE.</td>
</tr>
<tr>
<td>Agents</td>
<td>All agents</td>
</tr>
<tr>
<td>Agent Groups</td>
<td>All agent groups</td>
</tr>
<tr>
<td>Printers</td>
<td>All printers</td>
</tr>
<tr>
<td>Printer Groups</td>
<td>All printer groups</td>
</tr>
<tr>
<td>Queues</td>
<td>All queues</td>
</tr>
</tbody>
</table>
| Role Authorities | • Calendars  
                     | • Logins  
                     | • Agents  
                     | • Agent Groups  
                     | • Printers  
                     | • Printer Groups  
                     | • Queues |
| Substitution Variables | Substitution variables that ship with the product, for example #today, #yesterday, and #last_of_month. |
16.5 Sample Roles

16.5.2 Operator Roles

Operators need edit access to queues, thread schedules, agents and the Jobs/Explorer window to monitor the system. They also need read only access to several objects such as modules and chains so they can troubleshoot.

An operator’s key responsibility is monitoring system activity. To monitor the system, they need access to queues, agents, and the Jobs/Explorer window. Typically, an operator will not create or edit objects, but they may need to look at modules and chains to troubleshoot problems.

An operator should be able to access Explorer and edit:
- Queues
- Thread Schedules
- Agents

An operator should have read only access to:
- Modules
- Libraries
- Data Types
- Printers
- Chains
- Program Types
- Substitution Variables
- Applications
- Calendars
- Logins

Creating the Operator Edit Role

You should assign the following role authorities to a role with Edit authorization. This role could be named Operator_Edit.

- Queues
- Thread Schedules
- Agents
- Explorer

The operators will also need access to the objects they create including queues and thread schedules. However, access to these objects will be taken care of by assigning this role as the Maintenance role for each operator. When an operator creates one of these objects, it will automatically be assigned to this role.
Creating the Operator No-Edit Role

The table below shows the objects that should be assigned to the role without Edit authorization. This role could be named operator_no_edit.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendars</td>
<td>Calendars created by AppWorx administrators and operations.</td>
</tr>
<tr>
<td>Chains</td>
<td>All chains</td>
</tr>
<tr>
<td>Data Types</td>
<td>All data types</td>
</tr>
<tr>
<td>Logins</td>
<td>All logins</td>
</tr>
</tbody>
</table>
| Role Authorities  | • Modules
                   • Chains
                   • Applications
                   • Libraries
                   • Program Types
                   • Calendars
                   • Data Types
                   • Substitution Variables
                   • Logins
                   • Printers
                   • Explorer                                    |
| Modules           | All modules                                  |
| Agents            | All agents                                   |
| Printers          | All printers                                 |
| Substitution Variables | All substitution variables                  |

Default Maintenance Role

If you expect operators to create new queues or thread schedules, you will need to assign the Operator_Edit role as the Maintenance role for each operator. For more information on assigning a maintenance roles, see topic 16.2 Defining Users.
16.5 Sample Roles

16.5.3 End User Roles

End users need read only access to the Requests window and the Output window, as well as any modules they are expected to run.

End users generally submit jobs using the Requests window and view the output from the Output window. In some cases, you may want them to monitor the progress of their modules from the Explorer window.

Unlike programmers and operators, end users need only a single role without Edit authorization.

Creating the End User Role

The table below shows the objects that should be assigned to the end user role. The role does not include Edit authorization. This role could be named end_user.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role Authorities</td>
<td>Outputs</td>
</tr>
<tr>
<td></td>
<td>Requests</td>
</tr>
<tr>
<td>Modules</td>
<td>All modules they are expected to run.</td>
</tr>
<tr>
<td>Printers</td>
<td>All printers they are expected to use. They need access to printers to submit modules.</td>
</tr>
</tbody>
</table>

Maintenance Role Not Needed

End users are not expected to create any new objects such as modules and chains. Therefore, they do not need to be assigned a Maintenance role. For more information on assigning maintenance roles, see topic 16.2 Defining Users.
16.6 Defining Logins

If a program run by a module requires access to a database or server, you must create an AppWorx login. The AppWorx login object protects the actual database or server login and password while giving users access to it.

If a program run by a module requires access to a database or server, you must create an AppWorx login. The AppWorx login object protects the actual database or server login and password while giving users access to it. AppWorx logins include the login name, password, type of login, login ID or network connect string or host IP address, and an encryption flag. The encryption flag is set by the system when you update or add a login. After you create a login, other users can assign it to modules and chain components.

![Figure A. The Login window.](image)

**Note:** AppWorx roles control access to logins. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To add a new login:

1. From the Logins selector window, click **New**.
   
   AppWorx opens the Logins window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A.
Table A. The Login window fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The login name. If you need to create two or more logins with the same name (for example, you use the same login name on two databases), you will need to distinguish the names from one another (login names must be unique within AppWorx). To create unique names for AppWorx, append an '@' character followed by any number of characters to the end of a login name. The '@' and characters after it will be stripped off before the name is passed to a module. For example, if Oracle databases running on two hosts (sun1 and dg1) both use the same login name, add '@sun1' and '@dg1' to the login names in order to distinguish them from one another: sqloper@sun1 and sqloper@dg1.</td>
</tr>
<tr>
<td>Password</td>
<td>The password.</td>
</tr>
<tr>
<td>Type</td>
<td>Helps identify different types of logins. The 'Host' type is for special logins typically used with the FTP module.</td>
</tr>
<tr>
<td>Login ID</td>
<td>Can only be used with a local database residing on the same machine as the AppWorx database. For Oracle, enter the SID. If you enter a login ID, leave the Connect string field blank.</td>
</tr>
<tr>
<td>Connect string</td>
<td>You must enter a connect string if the database resides on a different machine than AppWorx. You can also use a database string to identify a local database. For example, a SQL*Net connect string for an Oracle database. Use an IP Address for a host. If you use a connect string, you should leave the Login ID field blank.</td>
</tr>
<tr>
<td>Encrypted</td>
<td>AppWorx automatically encrypts the password and checks the Encrypted field when you click OK. You cannot change the Encrypted setting.</td>
</tr>
</tbody>
</table>

Changing the AppWorx Database Login

When AppWorx is installed, a database login is created by default for logging in to the AppWorx database account (the repository). The database login is usually named the same as the database account name you provide during the installation process.

If you are going to change the AppWorx database login, you must first change the login in the database.
# Working with Agents

17.1 Introduction to Working with Agents ................................................................. 404
17.2 Defining Agents ................................................................................................. 406
17.3 Inactivating Agents ......................................................................................... 408
17.4 Setting the Master’s Time Zone ....................................................................... 410
17.5 How Time Zone Shifts are Calculated .............................................................. 412
17.6 Defining Agent Groups and Assigning Agents to Them ................................. 414
17.7 Understanding Dynamic Load Balancing ......................................................... 416
17.8 Managing System Records .............................................................................. 418
17.1 Introduction to Working with Agents

An agent is an instance of AppWorx, agents are installed on each machine where jobs are executed. An agent can be a master's local agent, or a remote agent. The master schedules and controls job execution on all the agents assigned to it. Agent groups distribute the load between agents on one or more machines.

With AppWorx masters and agents, you can control jobs on multiple machines from any AppWorx client (see Figure A). The AppWorx master resides on a server and launches local jobs via its local agent. During install, AppWorx creates the master and its local agent. Remote agents reside on other servers but receive commands via the master. For the master/agent relationship to work, AppWorx agent software must be installed on all the machines you wish to control. The AppWorx database must reside on a single machine.

**Figure A.** An AppWorx architecture diagram.

**How AppWorx Runs Jobs**

When the master determines that a job should be executed, it submits it to an AppWorx queue. The agents poll the queue. When an agent finds a job that it should run, it pulls that information from the queue and runs the job. When the job completes, it sends status information back to the master.
What to Consider When Planning Master/Agent Installations

A master controls and monitors the execution of jobs on its agents. This gives you centralized control over several machines. You can have any combination of masters and agents. For example, an agent of one master can reside on a machine containing another master.

When determining the number of masters and the number of agents assigned to each master, you should consider the reliability of your machines and network. If the master machine goes down, jobs will not be launched on any of the master’s agents. If a master loses network communication with an agent, jobs will not be launched on the agent. However, when the master is operational again, or network communications are restored, AppWorx will resume operations at the point they were interrupted.

Basic Steps for Installing an AppWorx Remote Agent

The basic steps for installing a remote agent are:

1. Define the remote agent object in AppWorx.
2. Create a user login.
3. Install the AppWorx remote agent software.
4. Start the Network Listener Service (for Windows).
5. Verify the installation.

Step 1 is described in topic 17.2 Defining Agents. Steps 2-5 are described in the AppWorx Installation Guide.

Using Agent Groups

Agent groups distribute the load between agents on one or more machines. You create agent groups and assign local and remote agents using the following guidelines:

- All agents must be connected to the same network.
- One or more agents can be assigned to a group.
- Agents can be assigned to more than one group.
- Agents in a group can be, and usually are, on different servers.
- You may have more than one agent on a server.

Figure B shows the relationships between agents and groups.

You specify an agent or agent group when you define a module. When you specify an agent or agent group for a module, you are telling AppWorx where the program is located and where it will be run. AppWorx will run the program on the first available agent in an agent group.
17.2 Defining Agents

You can add remote agents by adding an agent object in AppWorx and running an install script. Remote agents are used to run jobs on a machine that does not include the AppWorx master.

During install, AppWorx creates a master and its local agent. You can add remote agents by adding an agent object in AppWorx and running an install script. An AppWorx remote agent is used to run jobs on a machine that does not include the AppWorx master. You can edit remote agents and the master’s local agent from the AppWorx GUI.

Note: AppWorx roles control access to agents. If you do not have access to them, see your AppWorx administrator.

Adding Remote Agents

To add a remote agent object:

1. From the Agents selector window, click **New**.
   
   AppWorx opens the Select agent type window shown in Figure B.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Select **Standard** to define an AppWorx remote agent.
   
   For information on defining an OAE agent, see the Oracle Applications Extension Guide. For information on defining an SAP agent, see the mySAP.com Extension Guide.

3. Complete the fields using the information in Table A.

Table A. Agent window fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name can be up to eight characters long.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of an agent can be up to 30 characters long.</td>
</tr>
<tr>
<td>IP address</td>
<td>The IP address should be in the form <code>nnn.nnn.nnn.nnn</code>. You can use an alias, but it must be defined on the master and agents using the alias.</td>
</tr>
</tbody>
</table>
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep time</td>
<td>Defines how many seconds (10 to 1,000) AppWorx will wait between processing cycles if not currently processing.</td>
</tr>
<tr>
<td>Type</td>
<td>This non-edit field specifies the type of agent you are installing (Oracle Applications, SAP, or AppWorx).</td>
</tr>
</tbody>
</table>
| Thread schedule        | Sets the maximum number of concurrent jobs that can run on the agent at one time in all queues. For information on defining thread schedules, see topic 18.3 Defining Thread Schedules.  
**Note:** Thread schedules can be reassigned to agents (or the master) from the Jobs/Explorer window. For more information, see topic 5.12 Managing Agents. |
| Login type             | Displays the login type used when an operator runs a module on an ad hoc basis. This field’s value can be set for all agents from the master’s local agent.  
- Modules: The job runs under the module’s login (unless the module definition calls for the user’s login to be used). For more information, see topic 8.5 Specifying Output Options for Modules.  
- Users: The job runs under the user’s login (unless the user definition is set to ‘No selection’, in this case AppWorx will use the module’s login). This option is useful if users have different views of a database. For more information, see topic 16.2 Defining Users. |
| OS type                | Indicates whether the operating system used on the agent is UNIX or Windows. |
| Output directory       | Specifies the output directory where the system and program output from a module will be stored if an Output directory is not specified for the module. The default out directory will be displayed when you select an OS type. |
| Arguments              | This field is used for:  
- Inactivating the agent. For more information, see topic 17.3 Inactivating Agents.  
- Troubleshooting the agent. When necessary, AppWorx Technical Support will provide details on how to use this field. |
| User name              | The user name selected determines the Awexe range assigned to the agent. For information on setting the Awexe range, see topic 16.2 Defining Users.  
**Note:** Users assigned to a remote agent require an Awexe range for the agent to start. |
| Time zone              | This field is grayed out when adding remote agents. You can select a time zone for the master by editing this field for its local agent. For more information on the time zones, see topic 17.4 Setting the Master’s Time Zone. |
| Use module max revisions | When you define a module, you can specify the Max # of revisions of the output from the module that AppWorx will save. Select this option to use that setting. For more information, see topic 8.5 Specifying Output Options for Modules. |
| Master                 | This non-edit check box identifies the agent as the master’s local agent. |
17.3 Inactivating Agents

To skip all jobs on a particular agent, type 'INACTIVE' in the agent’s Arguments field.

To skip all jobs that run on an agent, type ‘INACTIVE’ in the agent’s Arguments field (see Figure A). To set jobs to run normally again, simply clear the Arguments field.

The status of the agent will be INACTIVE. All newly submitted jobs to that agent will go into the History with a status of SKIPPED. Jobs already submitted to the agent will continue as usual.

In Figure B, the PROD2 agent is inactive. One module and two chains that were skipped are displayed in the History.

Figure A. Enter ‘INACTIVE’ in an agent’s Arguments field to skip all jobs.

Figure B. Jobs assigned to an inactive agent will move to the History with a SKIPPED status.
17.4 Setting the Master’s Time Zone

To enable the Time Zone field for schedules, you must first select a time zone for the master (defined for its local agent). To add a time zone to the master, the master and database machine(s) both must use time zones and their clocks must be synchronized to within 15 minutes.

You may have jobs that you want to run at a particular time in a different time zone. You do this by making a selection from the Time zone field on the Schedule tab for a module or chain. To do this, you have to select the time zone for the master, so AppWorx will know how many hours to convert its start time.

Prerequisites

The following prerequisites must be met before setting the master’s time zone:

- The master and database must be installed on machine(s) using time zones.
- Both machines’ clocks must be set to the same time zone.
- Both machines’ clocks must be synchronized to within 15 minutes of one another.
Procedure

To set the time zone for the master.

1. Select the master’s local agent.
   The SUN26X agent has been selected in Figure A.

2. Select a time zone from the **Time zone** field.
   Select the standard time option if your computer does not adhere to daylight savings time. For example PST for Pacific Standard Time.
   UTC refers to the Universal Time Coordinate, which is used for the synchronization of computers on the Internet. For information on selecting time zones, see topic 17.5 *How Time Zone Shifts are Calculated*.

**Warning!** If you change the master's time zone from a value to ‘No selection’, any chains with time zones set will lose their time zone settings.
17.5 How Time Zone Shifts are Calculated

The table below lists all defined time zones, showing when shifts occur and the method used to determine them.

In the table below, the numbers in the Method column stand for the following:

- 0 – Standard Time with no changes.
- 1 – The last Sunday in October at 2 A.M., the first Sunday in April at 2 A.M.
- 2 – The last Sunday in October at 1 A.M., the last Sunday in March at 1 A.M.

<table>
<thead>
<tr>
<th>Time zone</th>
<th>Description</th>
<th>Method</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>JST</td>
<td>UTC+09 Japan</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>CCT</td>
<td>UTC+08 China Coast</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>WAST</td>
<td>UTC+07 West Australian</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>ZP6</td>
<td>UTC+06 Chesapeake Bay</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>EET</td>
<td>UTC+02 Eastern European</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CET</td>
<td>UTC+01 Central European</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>MEST</td>
<td>UTC+01(D2) Central European</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>UTC</td>
<td>UTC/GMT Greenwich Mean</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BST</td>
<td>UTC(D2) British Summary</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>WAT</td>
<td>UTC-01 W. Africa/Cape Verde</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>AT</td>
<td>UTC-02 Azores Std. Time</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>C</td>
<td>UTC-03 Brazil Std. Time</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>AST</td>
<td>UTC-04 Atlantic Std. Time</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>ADT</td>
<td>UTC-04(D1) Atlantic DST</td>
<td>1</td>
<td>-4</td>
</tr>
<tr>
<td>EST</td>
<td>UTC-05 Eastern Std. Time</td>
<td>0</td>
<td>-5</td>
</tr>
<tr>
<td>EDT</td>
<td>UTC-05(D1) Eastern DST</td>
<td>1</td>
<td>-5</td>
</tr>
<tr>
<td>CST</td>
<td>UTC-06 Central Std. Time</td>
<td>0</td>
<td>-6</td>
</tr>
<tr>
<td>CDT</td>
<td>UTC-06(D1) Central DST</td>
<td>1</td>
<td>-6</td>
</tr>
<tr>
<td>MST</td>
<td>UTC-07 Mountain Std. Time</td>
<td>0</td>
<td>-7</td>
</tr>
<tr>
<td>MDT</td>
<td>UTC-07(D1) Mountain DST</td>
<td>1</td>
<td>-7</td>
</tr>
<tr>
<td>PST</td>
<td>UTC-08 Pacific Std. Time</td>
<td>0</td>
<td>-8</td>
</tr>
<tr>
<td>PDT</td>
<td>UTC-08(D1) Pacific DST</td>
<td>1</td>
<td>-8</td>
</tr>
<tr>
<td>YST</td>
<td>UTC-09 Yukon Std. Time</td>
<td>0</td>
<td>-9</td>
</tr>
<tr>
<td>YDT</td>
<td>UTC-09(D1) Yukon DST</td>
<td>1</td>
<td>-9</td>
</tr>
<tr>
<td>AHT</td>
<td>UTC-10 Alaska/Hawaii Std.</td>
<td>0</td>
<td>-10</td>
</tr>
<tr>
<td>AHDT</td>
<td>UTC-10(D1) Alaska/Hawaii DST</td>
<td>1</td>
<td>-10</td>
</tr>
</tbody>
</table>
17.6 Defining Agent Groups and Assigning Agents to Them

To add an agent group, give the group a name and description. After creating agents and agent groups, you can assign agents to the agent groups. You can assign an agent to more than one agent group and assign priorities and factors for each agent.

Use agent groups to distribute the load on one or more machines. When you submit a job to an agent group, AppWorx runs the job on the first available machine in the group.

**Note:** AppWorx roles control access to agent groups. If you do not have access to them, see your AppWorx administrator.

**Procedure**

To create an agent group:

1. From the Agent Groups selector window, click **New**.
   
   AppWorx opens the Agent Groups window shown in Figure A.

   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Enter a name and description for the agent group.

3. If you want to run the jobs assigned to this agent group on every listed agent, select the **Multi-Execution** box.

   When you run a module through a multi-execution agent group, AppWorx automatically creates copies of the job that run on every agent in the group. Each job shows up in the Backlog so you can track execution.
4. Assign one or more agents to the agent group.
   To move a value from one table to the other, do one of the following:
   • Assign and unassign values by double-clicking them.
   • Highlight one or more values and select the arrow buttons to assign and unassign them.
     You can make multiple contiguous selections by holding down the Shift key and clicking with the left mouse button.
     You can make multiple noncontiguous selections by holding down Ctrl and clicking the left mouse button.
   • Use the double arrow buttons to assign or unassign all agents.

Setting the Priority and Factor for Agents in a Group

Dynamic Load Balancing provides the means to dynamically balance the computing load among several machines. The methodology used by AppWorx provides the flexibility to manage the distribution of load to suit the needs of almost any site.

Priority refers to the priority of the agent in comparison to the other agents in the agent group you have defined. Factor refers to the capacity of the agent in comparison to the other agents in the agent group you have defined.

Topic 17.7 Understanding Dynamic Load Balancing shows examples of how priorities and factors are used to balance the computing load across several machines.

To set an agent’s priority and factor:
1. Select an assigned agent from the Agent Group window.
2. Enter the agent’s priority and/or factor.
   The lower the number you assign to an agent in the Priority field, the higher priority it will have to run jobs.
   The lower the number you assign to an agent in the Factor field, the higher capacity it will have to run jobs. For example, if Machine A has a capacity of 20 and Machine B has half the capacity of Machine A, you would set Machine B’s factor to 40.
3. Click Update.
   AppWorx updates the Assigned agents display.
Dynamic Load Balancing provides the means to dynamically balance the computing load among several machines. The methodology used by AppWorx provides the flexibility to manage the distribution of load to suit the needs of almost any site.

Two fields you set for each agent in a group control the DLB:

- **Priority**: The priority of the agent in comparison to the other agents in the agent group.
- **Factor**: The capacity of the agent in comparison to the other agents in the agent group.

These fields are set in the Agent Group window. For information on assigning priorities and factors, see topic 17.6 Defining Agent Groups and Assigning Agents to Them. A combination of these parameters is used to determine the optimum agent for executing the next job.

The algorithm to determine the agent order is based on the following criteria:

1. The priority of the agent
2. \( \frac{\text{Agent factor} \times \text{count of jobs currently running on the agent}}{\text{maximum threads of agent}} \)
3. Last activity timestamp of the agent

After the algorithm is run, the agents are selected in ascending order.

**Examples**

**Example 1.** You want to evenly distribute your workload over the three machines below.

To spread the workload over the three machines, set all the priorities and factors equally.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Priority</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>B</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

**Example 2.** You want to evenly distribute your workload over the three machines below.

Machine A has twice the capacity of Machine B and Machine C. By setting the factors below, the workload will be distributed based on the capacity of the machines.
Example 3. You want to evenly distribute the workload over the three machines below. Machine A and Machine B will carry the primary workload of the three machines. Machine A has three times more capacity than Machine B. You use Machine C when Machine A and B reach capacity.

<table>
<thead>
<tr>
<th>Machine</th>
<th>Priority</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>50</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>C</td>
<td>50</td>
<td>40</td>
</tr>
</tbody>
</table>

Example 4. All General Ledger jobs will run with Machine A as the primary machine, but use Machine B when Machine A reaches capacity or is unavailable. All MRP jobs will run with Machine B and C as the primary machines, but overflow should run on Machine A. Machines A, B, and C have the same capacity.

In this case, we define two agent groups: One for General Ledger jobs and one for MRP jobs. Then all General Ledger modules should be assigned to the GL Agent Group and all the MRP jobs assigned to the MRP Agent Group.

**General Ledger Agent Group**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Priority</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>B</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>C</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

**MRP Agent Group**

<table>
<thead>
<tr>
<th>Machine</th>
<th>Priority</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>B</td>
<td>40</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>40</td>
<td>50</td>
</tr>
</tbody>
</table>

Balancing Modules in Chains

When AppWorx runs a chain that is assigned to an agent group, each module in the chain is run on the next available agent, as opposed to the entire chain being run entirely on one agent. The threading of the modules in a chain applies even if they run on different agents.
17.8 Managing System Records

AppWorx manages the records on the master by running the SYSTEM chain each night at midnight.

The SYSTEM chain is critical to the daily maintenance of AppWorx. It contains three modules: DELDEFAULT, HISTORY_PURGE, and RMI_LOG_ROLLOVER. The chain definition is shown in Figure A.

![Figure A. The SYSTEM chain.](image)

On a new install, the start time for the SYSTEM chain is set to run at 00:00 (12:00 midnight). You may wish to change this setting to a different time when processing is light.

Usually, you rely on the SYSTEM chain to maintain the history records, output files, and RMI logs. However, if you want to override the chain, you can run each module on an ad hoc basis.
Deleting History Records

The HISTORY_PURGE module deletes all history records that are older than the number of days you specify in the module's prompt. The default is 60 days. Figure B shows the module displayed in the Submit window.

![Figure B. You can set the number of days that history records will be kept.]

Deleting Output Files

When you define a module, you can set the number of days the output will be retained by AppWorx. The SODELETE module (alias DELDEFAULT) deletes the output files that have exceeded their retention settings.

Cycling RMI Log Files

RMI server logs list when users log into and out of AppWorx. By running the RMI_LOG_ROLLOVER module, you control the RMI server log's rollover interval. RMI logs are retained in the log folder for seven days. They are named rmiserver.<data and time>.log.

The date and time for the file name RmiServer.22092202.log translate to:
- Day of the month (22).
- Hour (09).
- Minute (22).
- Second (02).
18

Administering Queues

18.1 Introduction to Administering Queues ................................................................. 422
18.2 Defining Queues .................................................................................................. 424
18.3 Defining Thread Schedules ................................................................................ 426
18.1 Introduction to Administering Queues

You control the flow of jobs to servers by using AppWorx queues. All jobs pass through an AppWorx queue to get to a server. You control queue throughput by assigning each queue to a thread schedule. You can define an unlimited number and type of queues.

Thread schedules control the number of concurrent jobs that can run through a queue. When you define a thread schedule, you specify the number of threads, the minimum threads, and the start and stop times for the schedule. A thread schedule can be divided into subschedules, letting you change the number of threads for different times of the day.

For example, you might assign only one thread from midnight to 6:00 A.M., four threads from 6:00 A.M. to 5:00 P.M., and 2 threads from 5:00 P.M. to midnight. This gives you the ability to fine tune workloads on your system. The Queue Summary displays the current schedule in the Threads column (see Figure A).

![Figure A. The Queue Summary on the Explorer window](image)

<table>
<thead>
<tr>
<th>Queue</th>
<th>Status</th>
<th>Priority</th>
<th>High</th>
<th>Avg</th>
<th>Low</th>
<th>Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>BATCH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Monitor</td>
</tr>
<tr>
<td>EXPRESS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Monitor</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Monitor</td>
</tr>
</tbody>
</table>

Figure B shows several different queues. Queue 1 has zero threads from midnight to 8:00 A.M., three threads from 8:00 A.M. to 4:00 P.M., and one thread from 4:00 P.M. to 23:59:59. Queue 2 has one thread from midnight to 8:00 A.M., and three threads for the rest of the day.
When Threads are Set to Zero

If the thread setting for a queue is zero, any jobs scheduled to run through that queue will not be launched. The jobs will be displayed in the Backlog with a status of NULL.

**Example:** You can set the threads to zero as a method for preventing end user submissions from overwhelming the systems during the working day. To do this, set up a queue with a schedule that includes zero threads from 7 A.M. to 6 P.M., and 10 threads at all other times. End users would be able to submit jobs into the queue during the normal business day, but the jobs would simply be held until 6 P.M. when the queue opens up. This way, all jobs submitted during one day could be run overnight and made available the following day.

Selecting Minimum Threads with Thread Schedules

You can assign a minimum thread value to a thread schedule. Any queue assigned to a thread schedule with be able to have at least that number of threads available in a ‘standby’ mode.

For example, QUEUE_A (shown in the table below) is guaranteed to have 2 available threads regardless of any jobs requested or scheduled in other queues. If the total threads for the agent is set to 10, that would mean a maximum of 8 jobs could run at any time excluding QUEUE_A.

<table>
<thead>
<tr>
<th>Queue</th>
<th>Priority</th>
<th>Max threads</th>
<th>Min threads</th>
<th>Cumulative</th>
<th>Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE_A</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>QUEUE_B</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUEUE_C</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>QUEUE_D</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QUEUE_E</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>QUEUE_F</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Queues are grouped and run by priority. For example, in the table above you have two queues in each of the three priority levels (1, 2, 3). If the Agent Thread limit is set to 10, and the minimum thread limit is set to 2 for Queue_A and 1 for Queue_B, that leaves only 7 threads available to queues with other priorities. Based on the queues used in the example table above, if Queue_F was the only queue running jobs, only 3 jobs would be allowed to run because 7 min threads with a higher priority level are already reserved.
18.2 Defining Queues

To define a queue, you must name the queue, assign it to a thread schedule, set its priority, and make the queue active or inactive.

You control the flow of jobs to servers by using AppWorx queues. All jobs pass through an AppWorx queue to get to a server. You control queue throughput by assigning each queue to a thread schedule.

Procedure

To add a queue:

1. From the Queues selector window, click **New**.
   
   AppWorx opens the Queues window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A.

Figure A. The Queues window

Note: AppWorx roles control access to queues. If you do not have access to them, see your AppWorx administrator.
### Table A. The Queues window fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name may be up to 20 characters long.</td>
</tr>
<tr>
<td>Schedule</td>
<td>The thread schedule controls the maximum number of concurrent jobs that can run through the queue at any given time. For more information, see topic 18.3 Defining Thread Schedules.</td>
</tr>
<tr>
<td>Priority</td>
<td>Defines the order in which queues are scanned for job initiation. Queues are scanned starting with the number 1.</td>
</tr>
<tr>
<td>Active</td>
<td>Click the Active check box to activate the queue. Active queues apply the thread settings. When you inactivate a queue, no jobs will be processed through it. The jobs will remain in the Backlog with a NULL status until the queue is activated.</td>
</tr>
</tbody>
</table>
18.3 Defining Thread Schedules

A thread schedule defines the number of concurrent jobs that can run through a queue during a specified period of time. You can select multiple stop times for a thread schedule to control the minimum and maximum thread limits allowed throughout the day.

A thread schedule defines the number of jobs that can run concurrently through a queue during a specified period of time. You can divide a thread schedule into several subschedules, covering different time periods during a day. For example, during normal work hours you may want to limit a queue to two concurrent jobs, then in the evening reset the queue to 10 concurrent jobs. A thread schedule must cover the full 24 hours in a day.

Note: AppWorx roles control access to thread schedules. If you do not have access to them, see your AppWorx administrator.

Procedure

To define a thread schedule:

1. From the Thread Schedules selector window, click **New**.
   
   AppWorx opens the Thread Schedules window shown in Figure A.
   
   For information on using selector windows, see topic 2.5 Adding, Editing, and Deleting AppWorx Objects.

2. Complete the fields using the information in Table A and click **Update**.
   
   When you enter or change a field value a red triangle is displayed on the General tab to signify unsaved changes. For more information, see topic 8.3 Updating Unsaved Changes.

3. If you entered a stop time other than 23:59:59, select the second entry and assign values to it. You can enter as many stop times as you wish to set different minimum and maximum values to your schedule throughout the day.
### Table A. Thread Schedules window fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Identifies the thread schedule. A subschedule specifies a specific number of concurrent jobs for a specific block of time.</td>
</tr>
<tr>
<td>Description</td>
<td>A description for the schedule (up to 30 characters long)</td>
</tr>
<tr>
<td>Start time</td>
<td>The value for this non-editable field is set by AppWorx. A thread schedule's first entry will be midnight (00:00). The start time for each successive entry is determined by the stop time of the previous entry.</td>
</tr>
<tr>
<td>Stop hour</td>
<td>Used to enter a time that you want the change the values defined in the Min thread and Max thread fields.</td>
</tr>
<tr>
<td>Stop minute</td>
<td></td>
</tr>
<tr>
<td>Min thread</td>
<td>The number of the master's threads you wish to reserve for a queue. The values can range between 0 and the maximum number of threads available on the master. If a queue is assigned to a thread schedule with one or more minimum threads, then even higher priority queues (queues with a lower number set in their Priority field) will have this many fewer threads available to them. This is even true when no jobs are running in the queue that is reserving the threads.</td>
</tr>
<tr>
<td>Max thread</td>
<td>The maximum number of jobs that can run concurrently in the queue. The value can be between 0 and 999,999. If you leave this field blank, AppWorx sets the threads to 1. If you set the value to 0, the queue will not accept any jobs during the time period specified. Note: There is a maximum limit set for that controls the total number of jobs that can run concurrently through all queues.</td>
</tr>
</tbody>
</table>
Chapter 18: Administering Queues
19

Exporting and Importing Objects

19.1 Introduction to Exporting and Importing Objects .................................................. 430
19.2 Export/Import Flowchart ....................................................................................... 432
19.3 Export/Import Terms ............................................................................................. 434
19.4 Building Export Lists ............................................................................................. 436
19.5 Saving and Exporting Lists ................................................................................... 438
19.6 Opening Imports and Mapping Objects ................................................................. 440
19.7 Adding, Editing, and Deleting Maps File Entries .................................................. 442
19.8 Example: Exporting a Module to Another AppWorx Instance .............................. 444
19.1 Introduction to Exporting and Importing Objects

When you want to move AppWorx objects from one system to another, you use the AppWorx Export and Import operations. When importing, you can map any objects, including objects that were not exported. You save the map file and can use it for other imports.

**Warning!** You cannot import between versions of AppWorx (for example from 4.5 to 5.1). Doing so is wholly unsupported by AppWorx and can lead to serious version conflicts.

Most corporations maintain separate development, test, and production instances. These instances can be local (residing on the same host) or remote. When you are ready to move AppWorx objects from one system to another, you can use the AppWorx Export and Import operations (see Figure A).

You can export all AppWorx objects except agents, agent groups, logins, and users. These objects must exist or be created on the target system.

When you export, you select modules, chains, and support objects (the objects that are referenced by the modules and chains you select) that can be transferred from one instance to another. The export produces an .exp file.

When you import, all objects in the .exp file must either be:

- Included in the import.
- Mapped to an object with the same name in the target instance.
- Mapped to an object with a different name in the target instance.
- Created in the target instance.

With this functionality, modules, data types, roles, and other objects are developed in one instance and transferred throughout the AppWorx environment. The source instance(s) (development and/or QA) contain objects that will be used in one or more additional instances.

In many cases, the objects on the development system will be different than on a test or production system. For example, a printer called DEV1 on the development machine may be replaced by TEST1 on the test machine and PROD1 on the production machine. To accommodate the different support objects, you can map objects when importing.

An initial export and import may require some effort to create the appropriate map. However, you can save the map and use it each time you perform an export-import between the same two machines. Subsequent export-imports may require few if any updates to the map.

The key steps for importing and exporting are described in topic 19.2 Export/Import Flowchart.
**Single vs. Multiple Exports/Imports**

The AppWorx Import/Export utilities provide flexibility when it comes to managing objects. In some cases, it may be practical to export only one object type at a time. In other cases, it may be practical to export a group of objects. These are organization-specific decisions.

**Figure A.** You can export and import objects from one machine to another.

**Copying AppWorx Instances**

The AppWorx export and import utilities are designed to move a limited number of objects. If you want to move or copy an entire AppWorx instance, you should use your database utilities. For information on moving and copying AppWorx instances, see chapter 6: *Copying and Moving AppWorx Instances* in the *AppWorx Technical Reference.*
19.2 Export/Import Flowchart

The flowchart presented in this topic details the actions involved in the export/import process.

The key steps in the export/import process are presented below and shown in the flowchart in Figure A:

- On the source instance, do the following:
  - Select the modules and chains you wish to export.
  - Add the support objects for the modules and chains with the References tab.
  - If you are doing a repetitive export, save the export list
  - Export creating an .exp file in the export folder under AW_HOME.
- FTP or copy the .exp file to the import folder under AW_HOME on the target system.
- On the target instance, do the following:
  - Load the .exp file by going to the File menu and selecting Open Import.
  - Map objects and/or select a saved map.
    If any support objects are missing map, re-import, or create them on the target instance.
  - If you are doing a repetitive import, save your map file
  - Import.
Select objects to export

Save the export list, creating a .dat file.

Export the list, creating an .exp file in the export folder

FTP/copy the .exp file to the import folder on the target

Load the .exp file

Use an existing map file?

Create objects on the target machine

Are they defined on the source?

Map any unmapped objects

Any required objects still missing?

Is this a repetitive export?

Yes

No

Yes

No

Yes

No

Save map file

Import

Select map file

Select additional objects to export

Is this going to be a repetitive export?

Yes

No

Figure A. Export/import flowchart
19.3 Export/Import Terms

The following terms are used in this guide when discussing exports and imports.

**Export file:** An .exp file which is created when an “export list” is exported. Export files are written to the export folder on the source instance.

**Export list:** The list of objects you select from the Export window. The objects in the export list are displayed on the Objects to be exported tab. As you build an export list, AppWorx determines the “referenced objects”. Export lists can be saved at any time. Export files are saved to the export folder and have a .dat extension.

**Import file:** An “export file” which has been copied or transferred from the export folder on the source instance to the import folder on the target instance.

**Map:** A matching of imported objects to existing objects on a target system. Maps can be made on the Import or Map File tab of the Import window.

**Map file:** A list of “mapped” objects used when importing. The mapped objects are displayed on the Map File tab when importing. Map files can be saved at any time. Map files are located in the map folder and have a .dat extension.

**Referenced objects:** Unselected objects that are assigned to the objects you select when building an export list. When exporting, you can select the References tab to view and add reference objects. Agents, agent groups, logins, and users are displayed as referenced objects, but cannot be added to the export list. These objects must exist or be created on the target instance.

**Support objects:** Objects that are assigned to the modules and chains you select when building an export list.
19.4 Building Export Lists

To export objects, go to the Operations menu and select Export. Use the Export window to add objects to an export list. You can export all objects except agents, agent groups, logins, and users.

You can export objects with the Export window shown in Figure A. As you build a list of export objects, AppWorx determines the support objects (for example: data types, printers, logins, roles) that must be present on the target system to run the modules and chains. You can review these items on the References tab and add them to the export list. Export lists can be saved at any time.

Note: The Export operation may not be available to you. If you need to use it, see your AppWorx administrator.

Procedure

To select objects to export:

1. Open the Export window shown in Figure A by going to the Operations menu and choosing Export.

2. Choose the type of object you wish to export.

   AppWorx displays all objects of the selected type on the Objects tab. In Figure A, (ALL) is selected, displaying all objects.

3. Move objects to the Objects to be exported tab (or back).

   For information on assigning options, see topic 2.2 Working in the AppWorx Windows.

Warning! If an imported object does not exist on the target instance, it is added to the target instance’s database. Objects that already exist are overwritten. You do not get a warning message.
Exportable Objects

All objects can be exported except agents, agent groups, logins, and users. You must uniquely specify any object(s) to be exported. For example, if you export modules or chains that use substitution variables, the variables will not automatically be exported. You must export substitution variables explicitly as their own objects. When you export roles, AppWorx does not include the objects that have been assigned to them.

**Note:** Do not export the DBA role, doing so will cause errors.

When you Import, AppWorx displays the referenced objects that are not included in the export. The referenced objects can be:

- Included in the import.
- Mapped to an object with the same name in the target instance.
- Mapped to an object with a different name in the target instance.
- Created in the target instance.

Reviewing References

As you add objects to the list, you can get a list of the referenced objects required by selecting the References tab (see Figure B). Use the arrow buttons to add these objects to the Objects to be exported tab. You can refresh the Reference tab by going to the References menu and selecting Refresh.

Exporting Chain Components

If you select a chain, AppWorx does not automatically export its chain components. Once you select the chain, click the Reference tab and add the referenced modules and chains you wish to export with. AppWorx does not automatically add the components, because they will override the definitions of the modules and chains on the target instance.

Next Step

After building your export list, you export. If you are doing a repetitive export you will want to save it first. These actions are described in topic 19.5 Saving and Exporting Lists.
19.5 Saving and Exporting Lists

Export lists can be saved at any time. To save an export list for use at a later time, go to the File menu and select Save. If you are doing a repetitive export it is a good idea to save your export file before exporting, this way you will have a record of the export. To export a list, select Export.

If you are doing a repetitive export—an export where you have selected objects that you will export multiple times, you will want to save your export list to use again. It is recommended that you save your export list immediately before running a repetitive export. If you are doing a one-time export there is no need to save the export list before exporting.

Figure A. Export lists can be saved at any time.

Saving Export Lists

To save an export list:

1. Go to the File menu and select Save.
   AppWorx displays the Save Export List window (see Figure A).

2. Enter a file name and click OK.
   The name may be up to 30 characters long. AppWorx stores the export list file in the export folder under AW_HOME with a .dat extension.
The export list (.dat) file saves the names of the selected objects but not the elements that are contained in the object. Export lists are convenient when performing repetitive exports, generally common early in the development and testing phases. Instead of continually recreating your export list every time you want to export an object, (or in most cases, a variety of objects) use the save command to save your updates. We recommend using a naming convention that reflects its contents (APP_Daily, or End_of_Month).

You can open an export list by going to the File menu and selecting Open.

**Exporting Lists**

To export a list of objects:

1. Click the **Export** button.
   
   AppWorx displays the Save Export File window shown in Figure B. If you have saved the export file, that name will be displayed in the **Name** field.

   ![Figure B. Save Export File window](image)

2. Enter a valid file name and click **OK**.
   
   The Export utility runs a module called EXPORT. When the module completes executing it writes the file to the export folder under **AW_HOME** with an .exp extension. You can view the output file from the History.

**Moving the Export File to the Target Instance**

Before you can perform an import, you must FTP/copy the export (.exp) file to the import folder under the **AW_HOME** directory on the target machine. When you move the export file to the import folder, it is referred to as an import file. They are the same file.

**Next Step**

Once you export and move the .exp file, you are ready to open the import file and map the objects. These actions are described in topic **19.6 Opening Imports and Mapping Objects**.
19.6 Opening Imports and Mapping Objects

To import objects, go to the Operations menu and select Import. To open an import file, go to the File menu and choose Open import. You can map objects from the source system to objects on the target system. When you import objects, all mapped objects (for example: queues, printers) must exist on the target machine.

**Warning!** You cannot import between versions of AppWorx (for example from 4.5 to 5.1). Doing so is wholly unsupported by AppWorx and can lead to serious version conflicts.

After creating the export file on the source instance and moving it to the target instance, you are ready to import the file. When you import, AppWorx displays the objects referenced in the import that were not included in the export.

The referenced objects must be handled in one of the following ways:

- Included in the import.
- Mapped to an object with the same name in the target instance.
- Mapped to an object with a different name in the target instance.
- Created in the target instance.

The Import window is shown in Figure A.

**Note:** The Import operation may not be available to you. If you need to use it, see your AppWorx administrator.

**Warning!** If an imported object does not exist on the target instance, it is added to the target instance's database. Objects that already exist are overwritten. You do not get a warning message.

**Procedure**

To select an import file and assign a map:

1. Before importing data, make sure no one is working in AppWorx and that no jobs are being processed.
2. From the Import window, click the **Import** button. 
   AppWorx opens the Files window shown in Figure B.
   In the Export utility, you created an .exp file in the export directory. When you move the file to the import directory on the target machine, it is referred to as an import file. They are the same file.

3. Select a file and click **OK**.
   When you open an import file, AppWorx looks for matching resources in the target database. If it finds a matching resource, it displays an entry in the map column for the object. If it doesn’t find a matching resource, it leaves the entry blank. In Figure A, JOB_TITLE is not mapped.

4. If you want to use a map file that you have saved, open it by going to the File menu and selecting Open Map.

5. For each of the unmapped items, select an object from the list box in the map column.
   All objects must be mapped before you can import. If a support object does not exist, you can create it on the target instance. All objects you map are shown on the Map File tab, and will be saved when you save the map file. For information on the Map File tab, see topic 19.7 Adding, Editing, and Deleting Maps File Entries.

6. Select new values from the MAP column for any objects you wish to reassign.
   You may wish to reassign the default mappings between instances. For example, you may have defined queue TEST1 on the source machine. On the target machine, you may want queue TEST1 reassigned to queue PROD1. You can use a map to specify the change. Modules and chains cannot be reassigned, by default they have the same name on the source and target machines. All objects you map are shown on the Map File tab, and will be saved when you save the map file. For information on the Map File tab, see topic 19.7 Adding, Editing, and Deleting Maps File Entries.
   From the File menu select Save Map and enter a name on the Save window to save the map.
   Map files can be saved at any time. They exist independently from any export or import files. However, when you perform an import, you do so by assigning the map file to the import.

7. To import the file after all objects have valid destination names, select the **Import** button.
   The Import utility runs a module called IMPORT. You can check on the status of the module from the Jobs/Explorer window. The module creates a log called IMPORT.log.
   You can view the log with the File Viewer.
19.7 Adding, Editing, and Deleting Maps File Entries

You edit maps independently from import files. To edit entries use the Map File tab.

The Map File tab shown in Figure A includes a table of mapped objects. You can save and open map file using the Edit menu. When you save a map file, it is these rows that are saved. The table includes:

- Objects that have been mapped on the Import tab.
- Objects that have been added using the Object Details box.

The map table can contain support objects that exist in both databases, but this is optional. The table does not contain modules or chains.

Maps are independent from import files and can be edited and saved without affecting them. Changes you make on the Map File tab will be immediately reflected on the Import tab. The mappings do not alter the import file, but are applied to it during the import process.

Adding Additional Map Entries

There may be times when you want to add an entry to a map file that was not included in the import file. You might do this when objects don’t exist at the time of this import, but will the next time you use this map file.

To add an entry:

1. In the Object Details box, select an object type from the Type list box.
2. Enter the name of the object.
3. Select the object on the target instance you want to map it to from the Map list box.
4. Click Add.

AppWorx adds the entry to the table and to the Import tab.

Updating Entries

To edit an entry:

1. Highlight the entry in the table that you wish to update.

   AppWorx displays the values for the entry in the Object Details box fields.

2. Edit the values of the fields and click Update.

   AppWorx updates the entry in the table and on the Import tab.
Deleting Entries

To delete an entry:

1. Highlight the entry in the table that you wish to update.
   AppWorx displays the values for the entry in the **Object Details** box fields.
2. Click **Update**.
   AppWorx removes the entry from the table. If the entry was included in the import file, it will map to its default value on the Import tab. If the entry was not included in the import file, it will be removed from the Import tab.
19.8 Example: Exporting a Module to Another AppWorx Instance

Below is an example of how a user can export and import between AppWorx instances.

Developer Pat Brown has created the EMP_DEPT_JOB module in the development instance (see Figure A) and wants to export it to the test instance. Keep in mind that the module references support objects.

To complete the export/import, Pat takes the following steps:

1. Using the Export operation, Pat starts building the export list by selecting the module (notice that Pat used the Search field to find the EMP_DEPT_JOB module, good job Pat).
When Pat adds the module, the label for the References tab turns red, indicating that the module includes support objects.

2. Pat selects the References tab, and sees two data types created for the module, and adds them to the export list. Pat doesn’t need to add support objects that are the same on the development and production instances.

3. To save the export file, Pat goes to the File menu and selects Save as, then enters EMP_MOD1. AppWorx saves the file EMP_MOD1.dat in the export folder of the development instance. Using the Export utility, Pat can open the export list at a later time.

4. To export the file, Pat selects the Export button and is prompted to select a name for the export file. Pat can export the file using the same or a different name than the export list. Pat uses the same name. AppWorx saves the EMP_MOD1.exp file in the export folder of the development instance and runs the EXPORT module in the Backlog.

5. Next, Pat switches to the Explorer tab to sees that the module has finished.

6. Pat transfers the export (.exp) file to the import directory on the target machine. Once the file is moved, it is referred to as an import file. They are the same file.
7. Pat logs on to the target production instance and opens the Import utility. From the Import window, Pat selects the **Import** button.

AppWorx automatically maps the referenced objects. If a referenced object did not exist in the new instance, AppWorx would leave the object unmapped. All objects must be handled in one of the following ways:

- Included in the import.
- Mapped to an object with the same name in the target instance.
- Mapped to an object with a different name in the target instance.
- Created in the target instance.

8. Pat accepts the default mappings and selects Import from the File menu to import the file. AppWorx runs the IMPORT module in the Backlog.

9. Next, Pat switches to the Explorer tab to see that the module has finished.

The module and data types are now included on the production instance.
Appendices

Appendix A: Master/Agent Status Values ................................................................. 450
Appendix B: Job Status Values ............................................................................. 452
Appendix C: Troubleshooting Printer Problems ................................................... 458
Appendix D: Regular Expression Tables ............................................................... 460
Appendix A: Master/Agent Status Values

The possible agent status values are listed, along with a description of the status, and where appropriate, a suggested course of action.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| BUSY     | **Description:** The master is processing the job and has not had time to update the status, or the master died abnormally.  
**Action:** Do the following:  
• Check the master’s log file for errors. If errors occur, troubleshoot accordingly.  
• Check to see if the master’s awmaster process is running.  
• Turn on debug on the All debug mask for the master process and check for errors in the log and see if you can reproduce the problem.  
• Contact AppWorx Support, let them know whether the awmaster process was running and have the debug logs ready to send.  
**Note:** When the AppWorx master is processing conditions, it stops in the cycle to evaluate each SQL statement. Therefore, it is important that you write concise SQL statements. If a SQL statement holds up the master’s evaluation cycle, it will go into a BUSY status. |
| CHECK LOG| **Description:** The master or an agent has an error.  
**Action:** Check the master or agent’s log file for detailed information. |
| EXITING  | **Description:** This is an interim status that changes to STOPPED.  
**Action:** No action required. |
| IDLED    | **Description:** The agent is running but not processing.  
**Action:** Right-click and select the Resume option to allow processing to continue. |
| NETDOWN  | **Description:** An agent is unable to connect to the remote agent’s awtcpd process.  
**Action:** Verify network communication between master and remote agent hosts. |
| RESUMING | **Description:** This is an interim status that changes to RUNNING.  
**Action:** No action required. |
| RUNNING  | **Description:** The task is presently executing on the system.  
**Action:** No action is required. |
| STARTING | **Description:** The agent has been initiated by the master and is being activated for execution. This is an interim status that should change quickly to RUNNING.  
**Action:** No action required. |
<table>
<thead>
<tr>
<th>Status</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOPPED</td>
<td><strong>Description:</strong> This state indicates that the agent is down.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> Issue the <code>startso</code> command to restart the master and its local agent.</td>
</tr>
<tr>
<td>TROUBLE</td>
<td><strong>Description:</strong> Set by the master when it decides that the agent is not doing its job (for example, the master has sent it a job to start 5 minutes ago, but the agent never started it). The <code>so_back_process.so_last_activity</code> has not been updated for the amount set in the <code>Sleep time</code> field + 60 seconds.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong></td>
</tr>
<tr>
<td></td>
<td>• Check to make sure the <code>awmaster</code> process is running.</td>
</tr>
<tr>
<td></td>
<td>• Master: Locked row in database. May need to run <code>listpids.sql</code> to find any hung Oracle processes and stop them.</td>
</tr>
<tr>
<td></td>
<td>• Agent: Verify that there is adequate disk space.</td>
</tr>
<tr>
<td></td>
<td>• Check the master and agent log files in AppWorx log directory.</td>
</tr>
<tr>
<td></td>
<td>Check file for errors and verify all parameters set correctly for the agent.</td>
</tr>
<tr>
<td></td>
<td>• Check system CPU and memory usage for other processes that might be impacting AppWorx operation.</td>
</tr>
</tbody>
</table>
Appendix B: Job Status Values

The possible job status values are listed, along with a description of the status, and where appropriate, a suggested course of action. Statuses for remote jobs will have an R- in front of them.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| [blank]  | **Description:** No status has been assigned to the task as yet. AppWorx has not processed the task to determine execution eligibility.  
**Action:** Check to see if the agent is running.                                           |
| ABORTD   | **Description:** The task has terminated in an unsuccessful manner on the agent. This is an interim status that will change almost immediately to ABORTED.  
**Action:** Check the module definition., and wait for the status to change to ABORTED.     |
| ABORTED  | **Description:** AppWorx has processed the task and it terminated unsuccessfully.  
**Action:** Check the system log for the module to see why it aborted.                         |
| BAD APPL | **Description:** The application specified for the task is incorrect.  
**Action:** Check the application specified for the module.                                    |
| BAD BATCH| **Description:** There is no defined application information for BATCH registered in AppWorx.  
**Action:** Define a BATCH application.                                                       |
| BAD CONDITN| **Description:** A SQL statement or check file BEFORE or AFTER condition has returned an error in the module.  
**Action:** Edit the module's conditions and/or their SQL statements.                       |
| BAD CONNECT| **Description:** The login and password combination that is specified for the task is in error.  
**Action:** Check the login and password assigned to the module.                              |
| BAD DATE PRM| **Description:** The date format required by the argument type is incompatible.  
**Action:** Check the dates entered for a prompt if one exists, or check the dates passed to a prompt or condition by a substitution variable. |
| BAD LOGIN | **Description:** The login specified for the job is not registered in AppWorx and, therefore, cannot be found.  
**Action:** There may be a database problem. Check with your DBA to make sure the correct Database ID is specified for the module. |
| BAD MJN  | **Description:** The module job number (MJN) indicated on this task is incorrect and does not match the currently registered module known to AppWorx.  
**Action:** Have your DBA check the AppWorx database to see if there is a bad entry in the so_job_table table. |
<table>
<thead>
<tr>
<th>Status</th>
<th>Description/Action</th>
</tr>
</thead>
</table>
| BAD MODULE      | **Description:** A bad module name was entered for a REQUEST MODULE action in a condition.  
|                 | **Action:** Check the condition that generated the entry in Backlog. Check the details for the job to find the name of the chain the module belongs to. |
| BAD NODE        | **Description:** The agent assigned to the module does not exist. This status will only occur if the agent was defined through a direct entry into the AppWorx database.  
|                 | **Action:** Select an agent for the module.                                        |
| BAD QUEUE       | **Description:** A condition with the CHANGE QUEUE action has been met. The designated alternate queue name, however, does not exist in AppWorx.  
|                 | **Action:** Edit the condition.                                                     |
| BAD SQL STMT    | **Description:** A SQL statement associated with a dynamic substitution variable used by this job is incorrect.  
|                 | **Action:** Dynamic substitution variables can be used in prompts and conditions. Check what substitution variables are used in the prompts and condition for the module or chain |
| BAD TYPE        | **Description:** The date type specified for the job is incorrect.  
|                 | **Action:** Check the data type definition assigned to the prompt.                  |
| CANCELED        | **Description:** A condition on this task, or on a previous task in the chain in which this task resides, has been met which dictates that it and the rest of the chain be canceled.  
|                 | **Action:** Check the conditions specified for the job.                             |
| CHAIN WAIT      | **Description:** A chain with the Single run option checked has been submitted two or more times, and the task is waiting for all tasks in the prior chains to complete.  
|                 | **Action:** Check to see if a previous task has aborted or has an unusually long run time. To find out what the previous task is, check the details for the job. |
| CONDITION WAIT  | **Description:** The task is waiting for a condition to be met before executing.  
|                 | **Action:** Check the conditions for the job.                                       |
| CONFLICT        | **Description:** This task has a registered conflict with another task in the job queue that is currently executing.  
|                 | **Action:** No action is required. The status will clear when the dependent task completes executing. |
| DATE PENDING    | **Description:** This task is waiting until its scheduled execution time has been reached.  
|                 | **Action:** No action required.                                                     |
## Status Descriptions

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB ERROR</strong></td>
<td><strong>Description:</strong> An internal database error has occurred while processing the task. AppWorx captures the database message in the log files.</td>
<td><strong>Action:</strong> Call AppWorx Technical Support for help resolving the problem.</td>
</tr>
<tr>
<td><strong>DEAD</strong></td>
<td><strong>Description:</strong> The process associated with this task can no longer be found on the host system. This happens if a process takes longer than five minutes to begin execution after initiation or if its status is still RUNNING at the time that the process disappears from the host system.</td>
<td><strong>Action:</strong> Check to see if the agent is running. Check to see if the job was killed from UNIX.</td>
</tr>
<tr>
<td><strong>DELAYED</strong></td>
<td><strong>Description:</strong> The job has a condition that has been met indicating that the it will be delayed by a certain time period. The task will be evaluated for execution again at the new start time indicated.</td>
<td><strong>Action:</strong> Check the conditions for the job.</td>
</tr>
<tr>
<td><strong>DELETED</strong></td>
<td><strong>Description:</strong> The task has been deleted from the Backlog by an operator.</td>
<td><strong>Action:</strong> Find out why the operator deleted the job and resubmit the job if necessary.</td>
</tr>
<tr>
<td><strong>DIED</strong></td>
<td><strong>Description:</strong> The process associated with this task can no longer be found on the host system. This happens if a process takes more than five minutes to begin execution after initiation. Usually there is a problem with the agent that prevents the job from executing. If the job's status is still RUNNING at the time that the process disappears from the host system, the job may have been killed from UNIX.</td>
<td><strong>Action:</strong> Check to see if the agent is running. Check to see if the job was killed from UNIX.</td>
</tr>
<tr>
<td><strong>ERRORS</strong></td>
<td><strong>Description:</strong> AppWorx detected an error condition. Further information about the source of the error is documented in the module name field.</td>
<td><strong>Action:</strong> Call AppWorx Technical Support to report the problem, and get help resolving the situation.</td>
</tr>
<tr>
<td><strong>EXPIRED</strong></td>
<td><strong>Description:</strong> This status is displayed for the SYSTEM module only and indicates the AppWorx key system has expired.</td>
<td><strong>Action:</strong> Call AppWorx to receive a new activation key for your product.</td>
</tr>
<tr>
<td><strong>FILE ERROR</strong></td>
<td><strong>Description:</strong> The tmp file could not be created.</td>
<td><strong>Action:</strong> Check that the disk is not full or for improper file or directory permissions.</td>
</tr>
<tr>
<td><strong>FINISHED</strong></td>
<td><strong>Description:</strong> The task has successfully completed its operation.</td>
<td><strong>Action:</strong> No action required.</td>
</tr>
<tr>
<td><strong>HOLD</strong></td>
<td><strong>Description:</strong> The task has had its execution delayed indefinitely by a condition with a HOLD TASK action, or the task in the Backlog has been put on hold.</td>
<td><strong>Action:</strong> The task will remain on hold until the status is changed by an action associated with another condition or by a user.</td>
</tr>
<tr>
<td>Status</td>
<td>Description/Action</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| HOST FAILURE      | **Description:** An error occurred when the host attempted to execute the command required to perform the task. This usually indicates a lack of process slots on the system or an inadequate subprocess limit for the AppWorx account.  
**Action:** Check with your AppWorx administrator.                                                                                     |
| INITIATED         | **Description:** Chains (as well as schedule modules) will go to INITIATED status once their chains have been inserted into the job queue. Modifying these can effect the total chain.  
**Action:** No action required.                                                                                                           |
| KILL              | **Description:** A request to terminate the task while it was executing on the host was issued by a user or by an action associated with a condition. This is an interim status that changes almost immediately to KILLED.  
**Action:** See the explanation for the KILLED status. If you wish, you can resubmit the job.                                            |
| KILLED            | **Description:** AppWorx has processed the task after a request for termination has been made for it. The task has therefore been removed from the host system.  
**Action:** If the job is still in the Backlog, you can resubmit it.                                                                            |
| MAXREQUESTS       | **Description:** A job has restarted 99 times ($jobid = ###.99) and cannot be restarted.                                                                  
**Action:** Determine reason why the job will not complete, or select the Restart once on abort option for the module.                        |
| NETDOWN           | **Description:** An agent is unable to connect to the remote agent’s sotcpd process.  
**Action:** Verify network communication between master and remote agent hosts.                                                                |
| NO PRIORITY       | **Description:** This job has been scheduled with a priority of zero. Since no priority has been assigned to the job, it cannot be executed until a priority is set.  
**Action:** To change the priority for the module.                                                                                    |
| NODE WAIT         | **Description:** The module is assigned to an agent that does not have enough threads available to process the job.  
**Action:** The job will be processed as soon as there is an available thread. If this status occurs often, you may want to change the number of threads assigned to the agent. |
<table>
<thead>
<tr>
<th>Status</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td><strong>Description:</strong> Four possibilities:</td>
</tr>
<tr>
<td></td>
<td>1. No status has been assigned to the task as yet. It has not been processed by the master to determine execution eligibility. If this is the case the status should change quickly to STARTING.</td>
</tr>
<tr>
<td></td>
<td>2. The task follows a task that is single-threaded and part of a chain. The task cannot start until the single-threaded task has completed executing.</td>
</tr>
<tr>
<td></td>
<td>3. When all agent threads are occupied and a module must wait for an agent thread to open.</td>
</tr>
<tr>
<td></td>
<td>4. A previous task in the chain has aborted. The aborted module has the block chain option set; therefore, this task must wait for the aborted task to clear the queue before it can start.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> No action is required for the first three possibilities. For the fourth, check the conditions specified for the job that aborted. Troubleshoot the task and restart or delete it if from the Backlog.</td>
</tr>
<tr>
<td>PRED WAIT</td>
<td><strong>Description:</strong> The job is waiting for a predecessor to be met before executing.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> Check the predecessors for the job.</td>
</tr>
<tr>
<td>RESCHEDULED</td>
<td><strong>Description:</strong> A condition with a RESCHEDULE TASK action has been met. The master will re-evaluate the job and change its status to DATE PENDING.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> No action required.</td>
</tr>
<tr>
<td>RUNNING</td>
<td><strong>Description:</strong> The task is presently executing on the system.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> No action is required.</td>
</tr>
<tr>
<td>SELF WAIT</td>
<td><strong>Description:</strong> The Single run option is set for the module, and there is another instance of the job running. The first must complete before the second instance will be initiated.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> No action required.</td>
</tr>
<tr>
<td>SKIPPED</td>
<td><strong>Description:</strong> The task will not be run because an action associated with a condition has specified that the task be skipped.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> Check the conditions for the module.</td>
</tr>
<tr>
<td>STARTED</td>
<td><strong>Description:</strong> This state is displayed only in the History. It is associated with a master or agent and indicates the date and time the particular it was last started.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> No action required.</td>
</tr>
<tr>
<td>STARTING</td>
<td><strong>Description:</strong> The task has been initiated by the master and is being activated for execution.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> No action required.</td>
</tr>
<tr>
<td>START ERROR</td>
<td><strong>Description:</strong> The process necessary to run the job could not be launched.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> Check that there are adequate system processes and disk space where AppWorx is installed. Also, check for adequate disk space in the /temp directory. Check ulimit to ensure that the AppWorx user account does not have a process limit assigned to it (UNIX only).</td>
</tr>
<tr>
<td>Status</td>
<td>Description/Action</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>START FAILED</td>
<td><strong>Description:</strong> The agent did not get a response indicating that the job was launched successfully (no Process ID was returned). <strong>Action:</strong> Possible error with pm file. Check that there are adequate system processes and disk space where AppWorx is installed. Also, check for adequate disk space in the /temp directory (UNIX only).</td>
</tr>
<tr>
<td>STOPPED</td>
<td><strong>Description:</strong> This state indicates the time period that a master or agent was down before it was restarted. <strong>Action:</strong> To attempt to restart the master and its local agent, issue the SOSTART command.</td>
</tr>
<tr>
<td>TIME-OUT</td>
<td><strong>Description:</strong> The task has taken longer than the maximum run-time period specified in the module definition or a maximum run-time period specified in a condition. This is an interim status that should change quickly to TIMEDOUT. <strong>Action:</strong> See TIMEDOUT for actions.</td>
</tr>
<tr>
<td>TIMEDOUT</td>
<td><strong>Description:</strong> The master has processed the task after it has exceeded its permitted run-time allotment. <strong>Action:</strong> Determine why the task was taking so long to run. If appropriate, change the maximum run time for the module, or change the job's conditions.</td>
</tr>
<tr>
<td>UNAVAILABLE</td>
<td><strong>Description:</strong> The required agent is not running. <strong>Action:</strong> Start the agent from the Agent Status.</td>
</tr>
<tr>
<td>WARNING</td>
<td><strong>Description:</strong> An AppWorx run-time extension changed the status of a module to WARNING. Usually this status is assigned to a job that has errors but has gone to completion. <strong>Action:</strong> Check the log for the job to see what errors occurred.</td>
</tr>
</tbody>
</table>
### Appendix C: Troubleshooting Printer Problems

If you are having trouble printing output from AppWorx, use the following table as a troubleshooting guide.

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Causes</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printing to most printers is successful. However, one or more printers do not print, and don’t give any error messages.</td>
<td>The <strong>Inactive</strong> radio button is selected in the printer’s Auto print options box.</td>
<td>Make sure the <strong>Active</strong> radio button is selected in the printer’s Auto print options box.</td>
</tr>
<tr>
<td></td>
<td>The interface script assigned to the printer’s spooler is not defined correctly.</td>
<td>1. Verify that the interface script has execute permissions for AppWorx.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. After the interface script to echo its command line to a log file rather than executing the command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example change <code>eval &quot;lp $* $file&quot;</code> to <code>echo &quot;lp $* $file&quot; &gt; $AW_HOME/exec/LPP.log</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Look at the log file (for example, <code>LPP.log</code>) to check that the syntax of the command is correct.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Execute the log file (<code>sh LPP.log</code>) to see if it prints the document.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If it prints, then the problem may be with AppWorx. Call AppWorx Technical Support.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the file does not print, then the problem is with your system.</td>
</tr>
<tr>
<td>An AppWorx error message states that printing was unsuccessful.</td>
<td></td>
<td>Record the details of the message, then call AppWorx Technical Support.</td>
</tr>
</tbody>
</table>
## Output files are not being created.

### Causes
Permissions on the output directory may not be set correctly.

### Action
Verify that AppWorx has write permission in the output directory.

### New files cannot be written to the output directory because:
- There is no free disk space
- There are too many files already in the directory.

### Action
Verify that AppWorx can write a file to the output directory. Free some disk space or clean out unneeded files as necessary.

### The job failed before it could generate any output.

### Action
Check the Backlog or History to see if the job aborted. Determine why the job aborted.

## Output files can be viewed from the Jobs/Explorer window, but they do not appear on the Output window.

### Causes
The output was generated with an output function of PRINT or STORE.

### Action
To view the output:
- Query the Output window for status codes PRINT and/or STORE.
To make the Output files viewable by default:
- Set the output function for the module to STORE.

### The SYSOUT printer was assigned to the module or chain.

### Action
Select a different printer.
Appendix D: Regular Expression Tables

The full syntax accepted by regular expressions is described below.

Unix regular expressions comprise a powerful pattern matching language. In AppWorx, regular expressions are allowed in Search fields on the administration selection windows and when defining file associations. The full syntax accepted by regular expressions is described below.

### Characters

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unicodeChar</td>
<td>Matches any identical Unicode character</td>
</tr>
<tr>
<td>\</td>
<td>Used to quote a meta-character (like &quot;)</td>
</tr>
<tr>
<td>\</td>
<td>Matches a single &quot;&quot; character</td>
</tr>
<tr>
<td>\0nnn</td>
<td>Matches a given octal character</td>
</tr>
<tr>
<td>\xhh</td>
<td>Matches a given 8-bit hexadecimal character</td>
</tr>
<tr>
<td>\uhhhh</td>
<td>Matches a given 16-bit hexadecimal character</td>
</tr>
<tr>
<td>\t</td>
<td>Matches an ASCII tab character</td>
</tr>
<tr>
<td>\n</td>
<td>Matches an ASCII newline character</td>
</tr>
<tr>
<td>\r</td>
<td>Matches an ASCII return character</td>
</tr>
<tr>
<td>\f</td>
<td>Matches an ASCII form feed character</td>
</tr>
</tbody>
</table>

### Character Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[abc]</td>
<td>Simple character class</td>
</tr>
<tr>
<td>[a-zA-Z]</td>
<td>Character class with ranges</td>
</tr>
<tr>
<td>[^abc]</td>
<td>Negated character class</td>
</tr>
</tbody>
</table>
Standard POSIX Character Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:alnum:]</td>
<td>Alphanumeric characters</td>
</tr>
<tr>
<td>[:alpha:]</td>
<td>Alphabetic characters</td>
</tr>
<tr>
<td>[:blank:]</td>
<td>Space and tab characters</td>
</tr>
<tr>
<td>[:cntrl:]</td>
<td>Control characters</td>
</tr>
<tr>
<td>[:digit:]</td>
<td>Numeric characters</td>
</tr>
<tr>
<td>[:graph:]</td>
<td>Characters that are printable and are also visible. (A space is printable, but not visible, while an ‘a’ is both)</td>
</tr>
<tr>
<td>[:lower:]</td>
<td>Lower-case alphabetic characters</td>
</tr>
<tr>
<td>[:print:]</td>
<td>Printable characters (characters that are not control characters)</td>
</tr>
<tr>
<td>[:punct:]</td>
<td>Punctuation characters (characters that are not letter, digits, control characters, or space characters)</td>
</tr>
<tr>
<td>[:space:]</td>
<td>Space characters (such as space and formfeed to name a few)</td>
</tr>
<tr>
<td>[:upper:]</td>
<td>Upper-case alphabetic characters</td>
</tr>
<tr>
<td>[:xdigit:]</td>
<td>Characters that are hexadecimal digits</td>
</tr>
</tbody>
</table>

Non-Standard POSIX-Style Character Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:javastart:]</td>
<td>Start of a Java identifier</td>
</tr>
<tr>
<td>[:javapart:]</td>
<td>Part of a Java identifier</td>
</tr>
</tbody>
</table>

Predefined Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Matches any character other than newline</td>
</tr>
<tr>
<td>\w</td>
<td>Matches a “word” character (alphanumeric plus “.”)</td>
</tr>
<tr>
<td>\W</td>
<td>Matches a non-word character</td>
</tr>
<tr>
<td>\s</td>
<td>Matches a whitespace character</td>
</tr>
<tr>
<td>\S</td>
<td>Matches a non-whitespace character</td>
</tr>
<tr>
<td>\d</td>
<td>Matches a digit character</td>
</tr>
<tr>
<td>\D</td>
<td>Matches a non-digit character</td>
</tr>
</tbody>
</table>
### Boundary Matchers

<table>
<thead>
<tr>
<th>Match</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Matches only at the beginning of a line</td>
</tr>
<tr>
<td>$</td>
<td>Matches only at the end of a line</td>
</tr>
<tr>
<td>\b</td>
<td>Matches only at a word boundary</td>
</tr>
<tr>
<td>\B</td>
<td>Matches only at a non-word boundary</td>
</tr>
</tbody>
</table>

### Greedy Closures

<table>
<thead>
<tr>
<th>Closure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*</td>
<td>Matches A 0 or more times (greedy)</td>
</tr>
<tr>
<td>A+</td>
<td>Matches A 1 or more times (greedy)</td>
</tr>
<tr>
<td>A?</td>
<td>Matches A 1 or 0 times (greedy)</td>
</tr>
<tr>
<td>A{n}</td>
<td>Matches A exactly n times (greedy)</td>
</tr>
<tr>
<td>A{n,}</td>
<td>Matches A at least n times (greedy)</td>
</tr>
<tr>
<td>A{n,m}</td>
<td>Matches A at least n but not more than m times (greedy)</td>
</tr>
</tbody>
</table>

### Reluctant Closures

<table>
<thead>
<tr>
<th>Closure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A*?</td>
<td>Matches A 0 or more times (reluctant)</td>
</tr>
<tr>
<td>A+?</td>
<td>Matches A 1 or more times (reluctant)</td>
</tr>
<tr>
<td>A??</td>
<td>Matches A 0 or 1 times (reluctant)</td>
</tr>
</tbody>
</table>

### Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Matches A followed by B</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>(A)</td>
<td>Used for subexpression grouping</td>
</tr>
</tbody>
</table>
## Backreferences

<table>
<thead>
<tr>
<th>Backreference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\1</td>
<td>Backreference to 1st parenthesized subexpression</td>
</tr>
<tr>
<td>\2</td>
<td>Backreference to 2nd parenthesized subexpression</td>
</tr>
<tr>
<td>\3</td>
<td>Backreference to 3rd parenthesized subexpression</td>
</tr>
<tr>
<td>\4</td>
<td>Backreference to 4th parenthesized subexpression</td>
</tr>
<tr>
<td>\5</td>
<td>Backreference to 5th parenthesized subexpression</td>
</tr>
<tr>
<td>\6</td>
<td>Backreference to 6th parenthesized subexpression</td>
</tr>
<tr>
<td>\7</td>
<td>Backreference to 7th parenthesized subexpression</td>
</tr>
<tr>
<td>\8</td>
<td>Backreference to 8th parenthesized subexpression</td>
</tr>
<tr>
<td>\9</td>
<td>Backreference to 9th parenthesized subexpression</td>
</tr>
</tbody>
</table>
Appendices
Index

-A-
Abort if no output option, setting for modules, 179
ABORT TASK, value for condition action, 314
aborted modules, resubmitting, 88
aborting module, with conditions, 332
About AppWorx window, viewing, 37
Action field
  using arguments with, 314
  values available for conditions, 314
action timing, setting for condition actions, 316
actions, taking multiple actions for a condition, 305
activating/inactivating
  agents, 408
  chain components, 197
  queues, 425
ad hoc modules and chains
  requesting, 44
  submitting, 46
adding
  applications, 168
  calendars, 236
  chains, 192
  components to a chain, 194
  conditions, 304
  data types, 256
  dynamic substitution variables, 350
  exceptions for a schedule, 232
  libraries, 170
  modules, 160
  operator logs for a job, 104
  predecessors, 286
  printer distribution lists, 366
  printer groups, 368
  printers, 364
  program types, 172
  prompts to a module or chain, 248
  queues, 424
  schedule modules, 222
  spoolers, 370
SQL statements to data types, 257
static substitution variables, 348
thread schedules, 426
users, 378
AFTER timing, setting for conditions, 308
agent groups
  AppWorx definition of, 7
  assigning agents to agent groups, 415
  dynamic load balancing, 416
  multi-execution agent groups, 414
  setting
    the factor for agents in an agent group, 415
    the priority for agents in an agent group, 415
agent status values (status and description/action), 450
Agent Summary, column descriptions, 107
agents
  {agent} replacement value, 320
  {node} replacement value, 321
  {operator} replacement value, 321
  activating/inactivating agents, 408
  adding thread schedules to agents, 426
  AppWorx definition of, 7
  assigning
    thread schedules to agents, 407
    to agents groups, 415
    to chain components, 200
    to chains, 193
    to modules, 165
    to schedules, 229
    users to agents, 407
  balancing chain components, across agents, 417
  changing thread schedules for agents and masters, 109
  configuring, 404
  definition of, 8
  dynamic load balancing, 416
  idling and resuming, 109
  log listings for agents, viewing, 106
  multi-execution agent groups, 414
querying
  the History for agents/agent groups, 113
  the Output window for agents, 59
resetting, 109
setting
  arguments for agents, 407
  the IP address for agents, 406
  the login type for agents, 407
  the master's time zone, 410
  the OS type for agents, 407
  the output directory for agents, 407
  the sleep time for agents, 407
  the Use module max revisions option for agents, 407
starting and stopping, 109
status, viewing, 107
taking a BUST or TROUBLE agent offline, 109
aliases
  {module} replacement value, 321
  displaying for chain components, 278
  propagating chain component aliases, 197
  setting for chain components, 196
All Chain Requestors user option, 381
All User Backlog Edits, user option, 381
Allow changes, setting for prompts, 253
alternate printer, setting, 365
alternative viewers, using, 64
applications
  {application} replacement value, 320
  adding, editing, and deleting, 168
  AppWorx definition of, 6
assigning
  to chains, 193
  to modules, 164
AppWorx
  command line interface, using, 140
  launching from a URL, 26
  with WebStart, 26
  monitoring and reporting, 8
  object storage, 8
  running jobs, 8
  security, 382
  Technical Support, contacting, 18
  Web site, connecting from AppWorx, 36
appworxj, launching AppWorx from the command line, 27
arguments, setting for agents, 407
assigned objects, viewing, 31
assigning
  a command path to a program type, 173
  a different output directory to a module, 179
  a file extension to a program type, 173
  a host command to a program type, 173
  a parameter format to a program type, 173
  agents
    to agent groups, 415
    to chain components, 200
    to chains, 193
    to modules, 165
    to schedules, 229
  an output function to a printer, 367
  an output function to chain components, 203
  an output function to modules, 178
  applications
    to chains, 193
    to modules, 164
  calendars to modules and chains, 231
  data types to prompts, 253
  default output directories to users, 379
  libraries to modules, 166
  logins
    to chain components, 200
    to modules, 181
  objects to roles, 386
  options (generic), 25
  paths to libraries, 170
  printer groups
    to modules, 178
    to printers, 365
  printers to modules, 178
  priority to queues, 425
  program options to modules, 179
  program types to modules, 166
  programs to a module, 166
  prompts to modules and chains, 252
  queues
    to chains, 193
    to modules, 176
    to schedules, 229
  roles to modules, 184
run and skip calendars to chain components, 199
skip calendars to modules and chains, 229
spoolers to printers, 365
thread schedules
to agents, 407
to queues, 425
users
to agents, 407
to chain components, 203
to printers, 367
to schedules, 229
associated file viewers, using, 64
auto print options, setting, 365
Auto start Explorer, user setting, 35
Ave run time option
setting for chains, 193
setting for modules, 177

-B-

Backlog
adding predecessors, 286
change status
hold, 84
kill, 86
change status, kill, 86
deleting modules, 90
hold status, putting a module on hold, 84
limiting the list of jobs by ad hoc requests, 81
resetting modules, 88
reviewing jobs waiting in each queue, 68
status values (status and description/ action), 452
using, 68
bad conditions, evaluating, 301
balancing
chains components across agents, 417
loads, 416
BEFORE timing, setting for conditions, 308

-C-
calendars
adding, editing, and deleting, 236
AppWorx definition of, 6
assigning
calendars to modules and chains, 231
run and skip calendars to chain components, 199
skip calendars to modules and chains, 229
CANCEL CHAIN, value for condition action, 314
capacity, setting the factor for agents in an agent group, 415
chain components, definition of, 8
Chain Flows
simulating, 212
using a graphical flow chart, 210
using the Gantt chart, 210
viewing for modules and chains, 210
chain ID, [chain_id] replacement value, 320
chain sequence number, [chain_seq] replacement value, 320
Chain Summaries
using a graphical flow chart, 126
using the Gantt chart, 124
viewing for modules and chains, 122
chains
activating and inactivating chain components, 197
adding
a schedule to a chain, 228
components to a chain, 194
conditions, 304
exceptions for a schedule, 232
list of values prompts, 262
multi select prompts, 264
predecessors, 286
prompts, 248
adding, editing, and deleting, 192
aliases
propagating chain component aliases, 197
setting for components, 196
AppWorx definition of, 6
assigning
agents to chain components, 200
agents to chains, 193
an output function to chain components, 203
applications to chains, 193
calendars to chains, 231
logins to chain components, 231
prompts to chains, 252
queues to chains, 193
run and skip calendars to chain components, 199
skip calendars to chains, 229
users to chain components, 203
balancing chain components across agents, 417
Chain Flows
  simulating, 212
  viewing, 210
Chain Summaries, viewing, 122
copying
  chains, 214
  prompts, 268
defining notes for modules and chains, 182
exporting, 436
forecasts, viewing, 130
Graphical Forecasts, viewing, 132
importing, 440
local substitution variables, creating, 356
maximum number of modules in a chain, 190
nesting
  maximum levels, 190
  within other chains, 190
overriding prompts in a schedule, 234
passing values through a chain, 270
querying
  the History for chains, 113
  the Output window for chains, 59
requesting, 44
responding to prompts in, 46
running chains, 190
scheduling
  chain components, 196
  Reschedule from (date) option, 231
selecting
  a quote character for prompts, 253
  Allow changes for prompts, 253
  Multi select for prompts, 253
  Upper case for prompts, 253
  Value required for prompts, 253
setting
  Ave run time option, 193
  conditions for chain components, 208
  copies option for chain components, 203
  execution options for chain components, 200
  execution options for chains, 192
  output options for chain components, 202
  predecessors for chain components, 206
  priority for chains, 193
  prompts for chain components, 204
  Single run option, 193
  the days chain components will run, 198
  the Default value option for prompts, 253
  the end date for a schedule, 229
  the Keep history records option, 193
  the link options, 201
  the Max length option for prompts, 253
  the minimum/maximum values for prompts, 253
  the next run date for a schedule, 229
  the next run time for a schedule, 229
  the setting the single/multi thread options for chain components, 201
  the start date for a schedule, 229
  the start times for a schedule, 228
  the Use module condition option for chain components, 201
  the Use request printer option for chain components, 201
  the Variable name option for prompts, 253
  time zones, 229
subchaining
  maximum levels, 190
  within other chains, 190
  threading with the Single run option, 193
CHANGE Q, value for condition action, 314
changing
  a user's password, 378
  column order, 24
  file viewer text size from the viewer, 61
  logins, 401
  parameters of modules in a queue, 92
  your password, 28
CHECK FILE
  checking if a file exists, 326
  value for conditions, selecting, 310
CHECK HISTORY
examine the History, 338
value for conditions, selecting, 310
CHECK PROCESS, value for conditions,
selecting, 311
Check SQL
option for data types, 258
option for substitution variables, 351
CHECK_FILE [file_line] replacement value,
321
checking
elapsed time with conditions, 324
for finished jobs, 330
for finished jobs with conditions, 330
if a file exists with conditions, 326
if a process is running with conditions, 328
the History for a job, 330
closing windows, 23
color, descriptions of the status bar, 74
column order, changing, 24
columns, sorting, 24
command line
interface, using, 140
launching AppWorx, 27
command path
{command_path} replacement value, 321
assigning to program types, 173
command, setting for spoolers, 371
conditions
Action field, using arguments with, 314
adding, editing, and deleting, 304
AFTER timing, setting, 308
BEFORE timing, setting, 308
checking for finished jobs, 330
DELETED timing, setting, 308
deleting in the Backlog, 98
DURING timing, setting, 308
DURING_WAIT variable, setting, 302
evaluating bad conditions, 301
qualifier value, 312
replacement values, using in conditions, 320
setting
for chain components, 208
timing for conditions, 308
static substitution variables, defining with
a SET SUBVAR action, 349
taking more than one action for a condition, 305
types available for conditions, 310
using to
abort a module, 332
check elapsed time, 324
check if a file exists, 326
check if process is running, 328
check the History, 338
time for execution sequence in a chain, 340
trigger two or more actions, 336
values available for conditions, 306
viewing for a job, 98
configuring, agents and masters, 404
conflicts, assigning module conflicts with
conditions, 301
Connect string, {net_connect} replacement
value, 321
contacting technical support, 18
Continuous Monitoring, user setting, 35
controlling execution sequence in a chain
with conditions, 340
copies, setting
for distribution lists, 367
for modules, 179
the copies option for chain components, 203
the Default copies option for modules, 179
the number of copies for spoolers, 371
copying
chains, 214
duplicating prompts from another module
or chain, 248
modules, 186
prompts, 268
crashes, how AppWorx recovers from, 116
Create Procedure SubVars, user option, 381
creating
default maintenance roles, 397
end user roles, 398
local substitution variables, for chains, 356
operator edit role, 396
operator no_edit role, 397
program edit role, 394
program no_edit role, 395
CURRENT QUEUE, value for conditions,
selecting, 311
Current session only, user setting, 35
CURRENT TIME, value for conditions,
selecting, 311
470 Index

-D-

data types
  adding SQL statements, 257
  adding, editing, and deleting, 256
  AppWorx definition of, 6
  assigning to prompts, 253
  checking SQL statements, 258
  setting
    Date input/output formats, 257
    Max length, 257
    Min/Max value options, 257
    Referential checking option, 257
    the Upper case option, 257
Date input/output formats setting for data types, 257
days, setting the days chain components will run, 198
decimal values, for job IDs, 79
Default copies option, setting for modules, 179
default output directory, assigning to a user, 379
default requestor, assigning to a chain components in the User field, 203
Default value
  prompts, responding to in a request, 46
  setting for prompts, 253
defining
  default maintenance role, 397
  end user roles, 398
  operator edit role, 396
  operator no_edit role, 397
  programmer edit role, 394
  programmer no_edit role, 395
  programs in modules, 166
  role authorities, 388
  roles, 384
  static substitution variables
    with a condition's SET SUBVAR action, 349
DELAY TASK, value for condition action, 314
DELETE SUBVAR, value for condition action, 314
DELETED timing, setting for conditions, 308
deleting
  applications, 168
  calendars, 236
  chains, 192
conditions, 304
conditions in the Backlog, 98
data types, 256
dynamic substitution variables, 350
history records, 419
libraries, 170
module definitions, 160
modules
  from the Backlog, 90
  with the edit menu, 90
output files, 419
predecessors, 286
printer distribution lists, 366
printer groups, 368
printers, 364
program types, 172
queues, 424
spoolers, 370
static substitution variables, 348
thread schedules, 426
deleting exceptions for a schedule, 232
Deny FTP Option, user option, 381
Deny Outmgr Hide, user option, 381
desktop, restoring the display when error dialogs are out of view, 23
details
  viewing for a job, 92
  viewing for a module after execution, 101
disabling
  Backlog queries, 113
  Output queries, 58
distribution lists
  adding printer distribution lists, 366
  setting
    the number of copies, 367
    the printer option, 367
DLB, dynamic load balancing, 416
DURING timing, setting for conditions, 308
DURING_WAIT variable, setting, 302
dynamic load balancing, with agent groups, 416
dynamic substitution variables
  adding, editing, and deleting, 350
  defining procedures, 352
  definition of, 346
-E-

editing
<table>
<thead>
<tr>
<th>Applications</th>
<th>168</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calendars</td>
<td>236</td>
</tr>
<tr>
<td>Conditions</td>
<td>304</td>
</tr>
<tr>
<td>Data Types</td>
<td>256</td>
</tr>
<tr>
<td>Dynamic Substitution Variables</td>
<td>350</td>
</tr>
<tr>
<td>Exceptions for a Schedule</td>
<td>232</td>
</tr>
<tr>
<td>Import Maps</td>
<td>442</td>
</tr>
<tr>
<td>Libraries</td>
<td>170</td>
</tr>
<tr>
<td>Modules</td>
<td>160</td>
</tr>
<tr>
<td>Predecessors</td>
<td>286</td>
</tr>
<tr>
<td>Printer Distribution Lists</td>
<td>366</td>
</tr>
<tr>
<td>Printer Groups</td>
<td>368</td>
</tr>
<tr>
<td>Printers</td>
<td>364</td>
</tr>
<tr>
<td>Program Types</td>
<td>172</td>
</tr>
<tr>
<td>Queues</td>
<td>424</td>
</tr>
<tr>
<td>Spoolers</td>
<td>370</td>
</tr>
<tr>
<td>Static Substitution Variables</td>
<td>348</td>
</tr>
<tr>
<td>Thread Schedules</td>
<td>426</td>
</tr>
</tbody>
</table>

**Elapsed Time, Checking with Conditions** | 324 |
**Email, Output Devices, Adding** | 362 |
**Enable Explorer, User Setting** | 35 |
**End Date, Setting for Schedules** | 229 |
**Error Dialogs, Restoring the Display When Error Dialogs Are Out of View** | 23 |
**Exceptions, Adding, Editing, and Deleting for Schedules** | 232 |
**Execution Options, Setting for**
- Chain Components | 200 |
- Chains | 192 |
- Modules | 176 |
**Existing Scripts, Getting Started with** | 12 |
**Explorer, See Jobs** |
**Exporting, Modules and Chains** | 436 |

---

**-F-**

**Factor, Setting for an Agent in an Agent Group** | 415 |
**Fax, Output Devices, Adding** | 362 |
**File Associations, Adding** | 64 |
**File Extension, Assigning to Program Types** | 173 |
**File Versions, Viewing** | 37 |
**Filtering Jobs by Agent and/or Queue in the Backlog and History** | 114 |
**Finding Specific Text, Viewing Output** | 61 |
**Forcing Uppercase in Prompts with Data Types** | 257 |

**Forecasts, Viewing for Modules and Chains** | 130 |
**Function, Assigning to**
- Chain Components | 203 |
- Modules | 178 |
- Printers | 367 |

---

**-G-**

**Gantt Chart, Using with**
- Chain Flows | 210 |
- Chain Summaries | 124 |
- Graphical Forecasts | 132 |
**Generating, Production Schedule Reports** | 136 |
**GOTO CONDITION, Value for Condition Action** | 314 |
**Graphical Flow Chart, Using with**
- Chain Flows | 210 |
- Chain Summaries | 126 |
- Graphical Forecasts | 132 |
**Graphical Forecasts**
- Submitting the FORECAST Module | 134 |
- Using a Graphical Flow Chart | 132 |
- Using the Gantt Chart | 132 |
- Viewing for Modules and Chains | 132 |
**Grouping, Predecessor Statements** | 288 |

---

**-H-**

**Help, Viewing Online Help** | 36 |
**Hidden Jobs in the Output Window, Hiding and Displaying** | 56 |
**Hide RO Prompts, User Option** | 381 |
**History**
- Checking with Conditions | 338 |
- Querying | 112 |
- Reviewing Completed Jobs | 112 |
- Viewing Output | 101 |
**HISTORY_PURGE, Deleting History Records** | 419 |
**Hold Status in the Backlog** | 84 |
**HOLD TASK, Value for Condition Action** | 314 |
**Host Command**
- \{host\_command\} USER Replacement Value | 321 |
- Assigning to Program Types | 173 |

---

**-I-**

**Idling Agents** | 109 |
**Importing**
editing maps, 442
modules and chains, 440
selecting an import file and map, 440
inactivating/activating
agents, 408
queues, 425
IP address, setting for agents, 406

-J-
Job History
records, deleting, 419
status values (status and description/action), 452
Job History Limits, user setting, 35
job ID
[jobid] replacement value, 321
decimal values, 79
querying
the History for job IDs, 113
the Output window for job IDs, 59
job statuses, querying the History for job statuses, 113
Jobs
adding operator logs, 104
changing thread schedules for agents and masters, 109
filtering jobs by agent and/or queue in the Backlog and History, 114
killing jobs, 86
object icons, definition of, 71
object keys, definition of, 71
object tree, definition of, 71
querying the History, 112
resubmitting jobs, 88
viewing
job conditions, 98
job details, 92
job notes, 100
job predecessors, 96
operator logs, 104
prompts, 94

-K-
Keep history records option
setting for chains, 193
setting for modules, 177
keyboard navigation, 25
KILL TASK, value for condition action, 314
killing jobs, 86

-L-
launching AppWorx, 26
launching AppWorx, from the command line, 27
libraries
{application_path} replacement value, 320
adding, editing, and deleting, 170
AppWorx definition of, 6
assigning
paths to libraries, 170
to modules, 166
limiting the list of jobs in the Backlog by ad hoc requests, 81
link options, setting for chain components, 201
List of Values (LOV) prompts
adding, 262
responding to in a module, 48
load balancing, 416
local substitution variables, creating for a chain, 356
logging on, to AppWorx, under different user names and masters, 28
logical expressions
using to limit the jobs listed in the Backlog, 112
using to limit the jobs listed in the Output window, 58
login
{db_type} replacement value, 321
options, setting for modules, 180
login source options, setting for modules, 181
login type, setting for agents, 407
logins
{oracle_sid} replacement value, 321
AppWorx definition of, 7
assigning to chain components, 200
assigning to modules, 181
changing, 401
logs
viewing agent logs, 106
viewing and adding operator logs for a job, 104
LOV prompts, adding, 262
manuals, viewing online help, 36
mapping support objects for imports, 440
master
  {master} replacement value, 321
assigning to agent groups, 415
changing thread schedules, 109
configuring, 404
definition of, 8
dynamic load balancing, 416
logging on to a master, 28
setting
  priority for agents (including the master) in an agent group, 415
  the factor for agents (including the master) in an agent group, 415
time zones, 410
Max # of revisions option, setting for modules, 179
Max length
  setting for data types, 257
  setting for prompts, 253
Max run time option, setting for modules, 176
maximum/minimum values, setting for prompts, 253
Min run time option, setting for modules, 177
Min Threads, setting, 111
Min/Max values setting for data types, 257
minimum threads allowed for queues, setting, 111
minimum/maximum values, setting for prompts, 253
MODULE REQUESTED, value for conditions, selecting, 311
MODULE RUNNING, value for conditions, selecting, 311
Module/chain option, using with roles, 387
  {module} replacement value, 321
activating and inactivating chain components, 197
adding
  components to a chain, 194
  conditions, 304
  exceptions for a schedule, 232
  list of values prompts, 262
  multi select prompts, 264
  predecessors, 286
prompts, 248
  schedules to a module, 228
adding, editing, and deleting, 160
aliases
  propagating chain component aliases, 197
  setting in chains, 196
AppWorx definition of, 6
assigning
  a different output directory to a module, 179
  agents to chain components, 200
  agents to modules, 165
  an output function to chain components, 203
  an output function to modules, 178
  applications to modules, 164
  calendars to modules, 231
  libraries to modules, 166
  logins to chain components, 200
  logins to modules, 181
  module conflicts with conditions, 301
  printer groups to modules, 178
  printers to modules, 178
  program options to modules, 179
  program types to modules, 166
  programs to modules, 166
  prompts to modules, 252
  queues to modules, 176
  roles to modules, 184
  run and skip calendars to chain components, 199
  skip calendars to modules, 229
  users to chain components, 203
balancing chain components across agents, 417
Chain Summaries, viewing, 122
copying
  modules, 186
  prompts, 268
defining notes for modules and chains, 182
deleting, 90
exporting, 436
forecasts, viewing, 130
Graphical Forecasts, viewing, 132
how modules fit into AppWorx, 159
importing, 440
in a queue, changing parameters, 92
maximum number in a chain, 190
overriding prompts in a schedule, 234
passing values through a chain, 270
prompts for chain components, 204
querying
  the History for modules, 113
  the Output window for modules, 59
requesting, 44
responding to prompts in, 46
scheduling
  chain components, 196
  Reschedule from (date) option, 231
selecting
  a quote character for prompts, 253
  Allow changes for prompts, 253
  Multi select for prompts, 253
  Upper case for prompts, 253
  Value required for prompts, 253
sequence number, {mjn} replacement value, 321
setting
  Abort if no output option, 179
  Ave run time option, 177
  conditions for chain components, 208
  copies for chain components, 203
  Default copies option, 179
  execution options, 176
  execution options for chain components, 200
  Keep history records option, 177
  login options, 180
  login source options, 181
  Max # of revisions option, 179
  Max run time option, 176
  Min run time option, 177
  output options, 178
  output options for chain components, 202
  predecessors for chain components, 206
  Print system output logs option, 179
  priority for modules, 176
  program information for modules, 166
  Restart once on abort option, 177
  Retention days option, 179
  Single run option, 177
  Stay in queue on abort option, 177

the Default value option for prompts, 253
the end date for a schedule, 229
the link options for chain components, 201
the Max length option for prompts, 253
the minimum/maximum values for prompts, 253
the next run date for a schedule, 229
the next run time for a schedule, 229
the single/multi thread options for chain components, 201
the start date for a schedule, 229
the start times for a schedule, 228
the Use module condition option for chain components, 201
the Use request printer option for chain components, 201
the Variable name option for prompts, 253
time zones, 229
status, monitoring and managing, 75
submitting, 46
threading with the Single run option, 177
viewing
details after execution, 101
output from History window, 101

monitoring and reporting, in AppWorx, 8
Multi Select (MS) prompts
adding, 264
responding to in a module or chain, 50
setting, 253
multi/single thread options, setting for chain components, 201
multi-execution agent groups, 414
multiple value prompts, responding to in a module or chain, 50

-N-
nesting chains within chains, 190
Next run date, setting for schedules, 229
Next run time, setting for schedules, 229
nodes, see agents
notes
  defining for modules and chains, 182
  viewing for a job, 100

numeric qualifiers, for conditions, 312
-O-

**OAE modules, adding**, 161
**object icons, definition of**, 71
**object keys, definition of**, 71
**object storage, in AppWorx**, 8
**object tree, definition of**, 71
**object-oriented approach to operations**, 4

**objects**
- adding, editing, and deleting, 30
  - assigning to roles, 386

**offline, taking a BUST or TROUBLE agent offline**, 109
**online manuals, viewing**, 36

**operator logs, viewing and adding for a job**, 104

**options**
- assigning (generic), 25
  - setting user options, 380

**organizing, roles**, 382

**OS type, setting for agents**, 407

**Output**
- bookmarking pages for printing, 61
- changing output formats, 61
- file viewer text size, changing, 61
- finding specific text, 61
- hiding jobs, 56
- limiting
  - list to jobs submitted by user, 57
  - to jobs submitted by user, 57
- previewing print jobs, 63
- printing
  - job output, 62
  - job output to an AppWorx system printer, 63
- querying the Output window, 58
- refreshing the display, 57
- using the scroll bars to view output, 61
- viewing
  - by page and line numbers, 60
  - from the Job History, 102
  - job output, 61

**output devices, adding**, 362

**output directory**
- assigning a different output directory to a module, 179
  - setting for agents, 407

**output files, deleting**, 419

**output function**
- assigning to chain components, 203
- assigning to modules, 178
- assigning to printers, 367
- querying the Output window for output functions (status), 59

**output options**
- setting for chain components, 202
- setting for modules, 178

**Output queries, logical expressions, using to shorten the jobs listed**, 59

**overriding prompts in a schedule**, 234

-P-

**parameter format, assigning to program types**, 173

**parameters, changing for modules in a queue**, 92

**password**
- changing for a user, 378
- changing your password, 28

**paths, assigning to libraries**, 170

**predecessors**
- adding, editing, and deleting, 286
- grouping predecessor statements, 288
- propagating chain component aliases, 197
- setting for chain components, 206
- viewing for a job, 96

**previewing print jobs**, 63

**print spoolers, AppWorx definition of**, 7

**Print system output logs option, setting for modules**, 179

**printer distribution lists, adding, editing, and deleting**, 366

**printer groups**
- adding, editing, and deleting, 286
- AppWorx definition of, 7
- assigning
  - to modules, 178
  - to printers, 365

**printer option**
- setting for distribution lists, 367
- setting for printers, 365
- setting for spoolers, 372

**printer, types, defining, See printer groups**

**printers**
- adding, editing, and deleting, 364
- AppWorx definition of, 7
- assigning
  - an output function to a printer, 367
printer groups to printers, 365
spoolers to printers, 365
to modules, 178
users to printers, 367
querying
the Output window for printers, 59
setting
auto print options, 365
print options for spoolers, 372
static options, 365
the printer option, 365
specifying default for a user, 379
troubleshooting, 458
printing
job output, 62
output, 62
output to an AppWorx system printer, 63
previewing print jobs, 63
the object tree or Explorer tables, 71
priority
assigning to queues, 425
setting
for an agent in an agent group, 415
for chains, 193
for modules, 176
for queues, 111
procedures, defining in dynamic substitution
variables, 352
production schedule
output abbreviations, 137
reports, running, 136
program information, setting for modules, 166
Program name, assigning to modules, 166
program options, assigning to modules, 179
program types
{command_type} replacement value, 321
adding, editing, and deleting, 172
AppWorx definition of, 6
assigning
a command path to a program type, 173
a file extension to a program type, 173
a host command to a program type, 173
a parameter format to a program type, 173
to modules, 166
commonly used AppWorx program types, 174
programs
{command} replacement value, 321
defining in modules, 166
prompts
types, defining, See data types
adding
list of values prompts, 262
multi select prompts, 264
to a module, 248
assigning
data types to prompts, 253
to modules and chains, 252
copying, 268
duplicating from another module or chain, 248
forcing uppercase with data types, 257
overriding prompts in a schedule, 234
responding to in a request, 46
responding to in modules and chains, 46
setting for chain components, 204
types, 248
viewing for a job, 94
propagating, chain component aliases, 197
putting a module on hold, 90
-Q-
qualifier value, for conditions, 312
Quemgr ALL Nodes, user option, 381
querying
disabling
Backlog queries, 113
Output queries, 58
the History
for agents/agent groups, 113
for chains, 113
for job IDs, 113
for job statuses, 113
for modules, 113
for queues, 113
for requestors, 113
the Output window
for agents, 59
for chains, 59
for job IDs, 59
for modules, 59
for output functions (status), 59
for printers, 59
for requestors, 59

Queue Summary, column descriptions, 111
queues
[batch_queue] replacement value, 320
activating/inactivating queues, 425
adding thread schedules to queues, 426
adding, editing, and deleting, 424
AppWorx definition of, 7
assigning
  priority to queues, 425
  thread schedules to queues, 425
to chains, 193
to modules, 176
to schedules, 229
changing parameters of modules waiting
  in, 92
prioritizing, 111
querying the History for queues, 113
reviewing status of, 110
Quote char, setting for prompts, 253

-R-
recovery
  after system crashes or outages, 227
  from system failures, 116
references
  referencing where objects are used, 30
  viewing where an object is used, 32
Referential checking, setting for data types, 257
refreshing the Output display, 57
regular expressions
  syntax for, 460
  using for file associations, 64
replacement values
  available in AppWorx, 320
  using in conditions, 320
  using the Subvar Replacement Values
    window, 354
REQUEST MODULE
  using arguments with the REQUEST
    MODULE action, 334
  value for condition action, 314
request options, See prompts
Request Queues, user option, 381
requestors
  assigning to schedules, 229
querying
  the History for requestors, 113
  the Output window for requestors, 59
Requests
  requesting modules and chains, 44
  submitting modules and chains, 46
Reschedule from, setting for schedules, 231
RESCHEDULE TASK, value for condition
  action, 315
resetting
  agents, 109
  modules, in the Backlog, 88
RESTART ON ABORT, value for condition
  action, 315
Restart once on abort option, setting for
  modules, 177
restoring the display, when error dialogs are
  out of view, 23
resubmitting aborted modules, 88
resuming agents, 109
Retention days option, setting for modules,
  179
RETURN CODE, value for conditions,
  selecting, 311
reviewing, jobs waiting in each queue in the
  Backlog window, 68
role authorities
  AppWorx definition of, 7
  defining, 388
roles
  AppWorx definition of, 7
  assigning
    objects to, 386
    to modules, 184
default maintenance role, creating, 397
defining, 384end user roles, creating, 398
general guidelines for creating, 392
module/chain option, setting, 387
operator edit role, creating, 396
operator no_edit role, creating, 397
organizing, 382
program edit role, creating, 394
program no_edit role, creating, 395
role authority defined, 388
RUN HOST COMMAND, value for condition
  action, 315
RUN TIME
  value for condition action, 315
value for conditions, selecting, 311

running
jobs in AppWorx, 8
production schedule reports, 136

-S-

SAP modules, adding, 161
schedule creep, avoiding with the Next run
date option, 231
schedule forecasts, viewing, 130
schedule modules
adding, 222
using, 220
schedules
adding
exceptions, 232
thread schedules, 426
thread schedules to queues, 425
to modules and chains, 228
assigning
agents to schedules, 229
queues to schedules, 229
users to schedules, 229
deleting for modules and chains, 228
editing for modules and chains, 228
overriding prompts in a schedule, 234
setting
the end date, 229
the next run date, 229
the next run time, 229
the start date, 229
the start times, 228
time zones, 229
scheduling
chain components, 196
Reschedule from (date) option, selecting, 231

selecting, import file and map, selecting, 440
selector windows, adding, editing, and
deleting objects, 30
SET ABORT STATUS, value for condition
action, 315
SET FINISHED STATUS, value for condition
action, 315
SET HOLD STATUS, value for condition
action, 315
SET SKIP STATUS, value for condition
action, 315
SET SUBVAR, value for condition action, 315

setting
a title or banner page for spoolers, 371
Abort if no output option for modules, 179
active option for queues, 425
aliases for chain components, 196
Allow changes for prompts, 253
arguments for agents, 407
auto print options, 365
Ave run time
for chains, 193
for modules, 177
conditions for chain components, 208
copies
for spoolers, 371
copies option for chain components, 203
Date input/output formats for data types, 257
Default copies option for modules, 179
execution options
for chain components, 200
for chains, 192
for modules, 176
INACTIVE argument for agents, 408
Keep history records option for modules, 177
link options for chain components, 201
login options for modules, 180
login source options for modules, 181
Max # of revisions option for modules, 179
Max length option for data types, 257
Max run time option for modules, 176
Min run time option for modules, 177
Min/Max option for data types, 257
Multi select for prompts, 253
multi-execution option for agent groups, 414
output options
for chain components, 202
for modules, 178
predecessors for chain components, 206
Print system output logs option for
modules, 179
printer options
for distribution lists, 367
for printers, 365
for spoolers, 372
priority
for agents in an agent group, 415
priority for chains, 193
priority for modules, 176
program information for modules, 166
prompts for chain components, 204
Referential checking option for data types, 257
Restart once on abort option for modules, 177
Retention days option for modules, 179
setting the single/multi thread options for chain components, 201
Single run option for chains, 193
Single run option for modules, 177
spooler commands, 371
static options for printers, 365
Stay in queue on abort option for modules, 177
the action timing for conditions, 316
the Default value for prompts, 253
the end date for a schedule, 229
the factor for agents in an agent group, 415
the IP address for agents, 406
the Keep history records option for chains, 193
the login type for agents, 407
the Max length for prompts, 253
the minimum/maximum values for prompts, 253
the next run date for a schedule, 229
the next run time for a schedule, 229
the number of copies for distribution lists, 367
the OS type for agents, 407
the output directory for agents, 407
the sleep time for agents, 407
the start date for a schedule, 229
the start times for a schedule, 228
the Use module max revisions option for agents, 407
the Variable name for prompts, 253
time zones, 229
timing for conditions, 308
Upper case
   for data types, 257
   for prompts, 253
Use module condition option for chain components, 201
Use request printer option for chain components, 201
user options, 380
Value required for prompts, 253
simulating, Chain Flows, 212
Single run option
   setting for chains, 193
   setting for modules, 177
single value prompts, responding to in a module, 48
single/multi thread options, setting for chain components, 201
SKIP TASK, value for condition action, 315
sleep time, setting for agents, 407
SODELETE, deleting output files, 419
sorting columns, 24
spoolers
   adding, editing, and deleting, 370
   AppWorx definition of, 7
   assigning to printers, 365
   setting
      a title or banner page, 371
      print options, 372
      the number of copies, 371
      the spooler command, 371
SQL statements
   adding to data types, 257
   checking for data types, 258
   checking for substitution variables, 351
standard modules, adding, 161
Start date, setting for schedules, 229
Start time(s), setting for schedules, 228
starting and stopping agents, 109
startso client
   launching AppWorx from the command line, 27
static options, setting for printers, 365
static substitution variables
   adding, editing, and deleting, 348
   definition of, 346
status
   agents, viewing, 107
   masters, viewing, 107
   modules, monitoring and managing, 75
status bar
   color descriptions of, 74
   viewing, 74
STATUS, value for conditions, selecting, 311
Stay in queue on abort option, setting for modules, 177
STAY IN QUEUE ON ABORT, value for condition action, 315
string qualifiers, for conditions, 312
subchainings within chains, 190
substitution variables
   AppWorx definition of, 6
   checking SQL statements, 351
   displaying aliases for chain components, 278
   dynamic, definition of, 346
   making local to a chain, 356
   passing values through a chain, 270
   static, definition of, 346
   types, 346
   using the Subvar Replacement Values window, 354
Subvar Replacement Values window, using, 354
support Web site, connecting from AppWorx, 36
SYSTEM chain
   running, 418
system crashes and outages, recovery, 227
system failures, how AppWorx handles, 116
-T-
Tail option for viewing output, 61
taking a module off hold, 85
taskbar, using, 23
technical support
   before you call, 18
   contacting, 18
   emergency (24 by 7), 18
   standard hours, 18
thread schedules
   adding, editing, and deleting, 426
   AppWorx definition of, 7
   assigning
      to agents, 407
      to queues, 425
   changing for agents and masters, 109
threading
   chains with the Single run option, 193
   how AppWorx 5.1 threads chain components, 216
   modules with the Single run option, 177
   setting the single/multi thread options for chain components, 201
TIME SINCE REQUESTED, value for conditions, selecting, 311
time zones
   setting for masters, 410
   setting for schedules, 229
timing, setting for conditions, 308
title setting a banner page for spoolers, 371
toolbar, using toolbar icons, 23
ToolTips, viewing and disabling, 22
triggering two or more actions with conditions, 336
TROUBLE, agent status value, 451
troubleshooting
   printers, 458
   setting arguments for agents, 407
types, available for conditions, 310
-U-
Upper case
   setting for data types, 257
   setting for prompts, 253
usage
   referencing where objects are used, 30
   viewing where an object is referenced, 32
Use database time zone, user setting, 35
Use module condition option, setting for chain components, 201
Use module max revisions option, setting for agents, 407
Use request printer option, setting for chain components, 201
USER DEFINED
   {condition1} replacement value, 321
   {condition2} replacement value, 321
   value for conditions, selecting, 310
user ID [requestor] replacement value, 321
user options
   All Chain Requestors, 381
   All User Backlog Edits, 381
   Create Procedure SubVars, 381
   Deny FTP Option, 381
   Deny Outmgr Hide, 381
   Hide RO Prompts, 381
   Quemgr ALL Nodes, 381
   Request Queues, 381
   View all User Outputs, 381
user settings
Auto start Explorer, 35
Continuous Monitoring, 35
Current session only, 35
Enable Explorer, 35
Job History Limits, 35
Use database time zone, 35

users
adding, 378
AppWorx definition of, 6
assigning
to agents, 407
to chain components, 203
to printers, 367
to schedules, 229
default output directory
assigning to, 379
 specifying, 379
logging on the AppWorx, 28
setting user options, 380
specifying default printers, 379

-V-
Value required, setting for prompts, 253
values
agent status (status and description/action), 450
available
for Action field for conditions, 314
for conditions, 306
Backlog (status and description/action), 452
Job History (status and description/action), 452

Variable name, setting for prompts, 253
View all User Outputs, user option, 381
view names, setting, See aliases
viewing
agent logs, 106
AppWorx file versions, 37
assigned objects, 31
Chain Flows, 210
Chain Summaries, 122
conditions for a job, 98
details for a job, 92
forecasts, 130
Graphical Forecasts, 132
notes for a job, 100
online manuals, 36
operator logs for a job, 104
output
 bookmarking pages for printing, 61
 changing output formats, 61
file viewer text size, changing, 61
finding specific text, 61
from the Job History window, 102
Tail option, 61
using the scroll bars to view output, 61
viewing by page and line numbers, 60
viewing the end of files, 61
predecessors for a job, 96
prompts for a job, 94
schedule forecasts, 130
the About AppWorx window, 37
the end of files, with the output viewer, 61
the status bar, 74
Tooltips, 22

-W-
WAIT UNTIL MET, value for condition
action, 315
WebStart, launching AppWorx, 26
where used, viewing where an object is
referenced, 32
windows, closing, 23