Getting Started with Prime Network Workflow

This chapter introduces Prime Network Workflow and its concepts and describes the taskflow for creating, testing, deploying, running, and viewing a workflow.

Topics include:

- **About Prime Network Workflow**, page 8-1
- **Workflow Concepts**, page 8-2
- **Taskflow for Creating a Workflow**, page 8-5

Additional information about Prime Network Workflow is available on the Cisco Developer Network (CDN)

### About Prime Network Workflow

Prime Network Workflow, which is based on LiquidBPM by Autonomy, Inc., is used to create and run the logical flows of activation commands, including complex rollback scenarios. This logic enables you to define relationships between tasks, including sequences, branches, failure procedures, and access to Prime Network commands as well as the information model. Prime Network Workflow can interface with an external system such as an order management system to create a full service-provisioning solution that is user-customizable and user-extendable.

Prime Network Workflow is a process management GUI that acts as a powerful visual design and execution tracing tool for defining and deploying activation workflows. A workflow consists of several tasks grouped and arranged in a hierarchy. Workflow management is supported in runtime, and includes a runtime GUI control console.

Prime Network Workflow is used to construct workflows which run gateway commands and provides complete access to the Prime Network live network information model. Prime Network Workflow provides a nested structure. Workflow commands are also available through the Prime Network API.

The workflow engine resides on the gateway using AVM 66. All deployed workflows are stored on the gateway. After a workflow is deployed, it is accessible through Prime Network Administration for viewing properties and status. Deployed workflows can be invoked through the Prime Network API using BQL. The workflow engine provides default workflow inherent rollback.

In addition, you can view a history of the invoked workflows using Prime Network Events.
Workflow Concepts

This section describes the following workflow concepts:

- Workflow Task, page 8-2
- BQL Task, page 8-3
- Lock/Unlock Task, page 8-4
- Workflow Call (Synchronous Workflow Nesting), page 8-4

Workflow Task

Tasks are added to workflows to define the processes. Each task performs a specific function. Tasks can be quickly added and configured using Prime Network Workflow.

Tasks can be classed as predecessor or successor tasks:

- A predecessor task is one that must be completed before the next task can be executed.
- A successor task is one that is run after a predecessor task.

For example, if a workflow consists of two tasks, Configure Switch (first task) and Configure Router (second task), the Configure Switch task is the predecessor task and must be completed before the Configure Router (successor task) can be run.

An activation state is associated with each task:

- Ready—The task is ready to begin when the constraints (for example, start time or predecessors) have been satisfied.
- Active—The task is being run.
- Done—The task is complete.
- Abort—The task has failed or the state has been set manually. The task can be manually reset to Ready or Done.
- Passive—The task exists but is no longer relevant. For the purposes of successive tasks, the passive task is considered done.

Figure 8-1 presents the typical task sequence.
BQL Task

The Execute BQL task is found in Prime Network Workflow toolbar. This task runs the BQL command specified in the Command Template tab of the Task Properties dialog box, and stores the result in a task attribute called Result so that it can be used by scripts and other tasks.
The Command Template tab can reference workflow attributes and task attributes. At runtime, the attribute’s values are substituted into the template before it is run.

**Note**

The BQL task in a workflow can only execute script BQLs and any of the generic commands: Get, Delete, Find, Update, Set.

**Note**

When you execute a workflow to run commands on a device, using the RunWorkflow BQL command, it will be executed only if the preview flag in the RunWorkflow BQL is false. If the value of the preview flag is true, it will only be previewed. See Running a Workflow, page 9-22 for details.

**Lock/Unlock Task**

The Lock task has two main objectives:

- To allow workflow instances to declare the resources that they use and the scope of their usage.
- To ensure that those resources are not used by any other workflow instance during that scope.

The Lock task enables you to protect any component from concurrent use by multiple workflows. You can lock an object that represents a single resource and guard access to it. A resource’s identifier serves as the name of the lock. At any given time, a lock can be owned by, at most, only one workflow. Resources can be automatically locked during rollback.

The system prevents deadlocks before they occur. Upon detecting an imminent deadlock, the lock operation belonging to the workflow with the least progress fails. A failed lock might or might not abort the workflow.

The locking mechanism does not cover every access to every resource. Only workflows participate in the locking process. Nonworkflow activities can access a resource even when it is locked by a workflow. Participation in the locking process is optional.

You can:

- Lock or unlock single or multiple resources.
- Unlock resources when a workflow terminates.
- Lock resources during rollback.

**Workflow Call (Synchronous Workflow Nesting)**

Synchronous workflow nesting allows workflow designers to invoke sub-workflows synchronously and pass arguments to each subworkflow invocation. This enables the workflow designer to isolate the tasks running in the subworkflow as much as possible from the tasks running in the parent workflow and in other subworkflows.

The following functionality is available:

- The child workflow is run in a separate workflow. The parent workflow waits for the child workflow to terminate.
- When a child workflow is stopped, it causes its parent workflow to stop as well.
• The child workflow has a separate scope for attributes.
• The output of the child workflow is directed to the parent workflow.
• The parent workflow can pass parameters to its child workflow.
• The correct rollback sequence is maintained throughout the depth of the lineage.
• Child workflows are not visible through the API. You interact directly with the parent.
• Delete and abort operations on parent workflows are delegated to child workflows.

Note  The maximum workflow nesting depth is defined in the registry. The default value is 16.

Taskflow for Creating a Workflow

Figure 8-2 presents the process required when working with and managing workflows. The tasks are described in the order in which they must be performed.
Step 1: Create a command using Command Builder and preview it

Step 2: Define tasks and workflows using Prime Network Workflow

Step 3a: Copy the command descriptor scripts

Step 3b: Edit the command descriptor scripts

Step 3c: Add workflow and task attributes

Step 4: Test the workflow locally

Step 5: Deploy the workflow on the gateway

Step 6: Run the workflow

Step 7: View the workflow in Prime Network Administration

Step 8: View the workflow results in Prime Network Events

1. Create a command using Command Builder and preview it. For more information, see Creating a Command, page 13-7.

2. Define tasks and workflows using Prime Network Workflow. For more information, see Chapter 9, “Working with Prime Network Workflow.”

3. Set up the command descriptor scripts and add workflow attributes:
   a. Copy the command descriptor scripts—Copy the command descriptor scripts that were defined using Command Builder to Prime Network Workflow BQL Tasks command template tabs.
   b. Edit the command descriptor scripts—Edit the command descriptor scripts that were defined using Command Builder to Prime Network Workflow BQL Tasks command template tabs, as required.
   c. Add workflow and task attributes—Add workflow and task attributes using Prime Network Workflow.

4. Test the workflow locally. For more information, see Testing a Workflow, page 9-20.

5. Deploy the workflow on the gateway. For more information, see Deploying a Workflow Template, page 9-5.
6. Run the workflow using a BQL command. For more information, see Running a Workflow, page 9-22.

7. Manage and view the workflow in Prime Network Administration. For more information, see Chapter 10, “Managing Workflows.”

8. View workflow results in Prime Network Events. Prime Network Events enables you to view the workflow history, including when workflows have been completed, their status, and the command invoked.

**Note**

For Prime Network Events to display an event when a Workflow is aborted, you must add the following line to the preActivateScript in the Callback Scripts tab:

```javascript
thisWorkflow.getRootTask().abort();
```

For more information, see Prime Network Workflow application online help for **Engine Behavior > Workflow State Transitions**.
Taskflow for Creating a Workflow