

Chapter 31

Object State Manager Functionality

The object state manager (OSM) is responsible for ensuring the transactional behavior of directory objects. This chapter contains the following sections:

- Object Life Cycle Management on page 507
- Transactions on page 507
- Creating a State Machine on page 512
- Locking a Workflow Transaction on page 513

Object Life Cycle Management

The OSM controls the life cycle of target objects. Thus it prevents the following:

- Duplicate requests
- Invalid requests

The OSM also allows for a customizable life cycle for each object instance; that is, objects of the same type can have different life cycles at the discretion of their creator.

Transactions

The OSM performs transactions on target objects. Each transaction has only one target. Transactions consist of an action and an optional side effect. The set of possible actions that a target object can be subjected to at each point in time is defined by a state machine, which is specified as a set of transitions. Every action can have an associated optional side effect, which, in this software release, is the name of a workflow to be executed.

Every target object is associated with one state machine and keeps track of the state it is in. If a requested action is valid—that is, if there is a transition from the current state of the target with a matching action on the associated state machine—a transaction is performed, which consists of simply changing the state of the target according to the given transition. If the transition specifies a side effect, a workflow is triggered, and the transaction is finished only when the workflow finishes; that is, the state of the target object is updated only after the workflow completion. If the workflow is successful, the transition that is guarded by the successful execution of the workflow is chosen. If it fails, the one guarded by the failure condition is then chosen. In both cases, the target has its state updated.

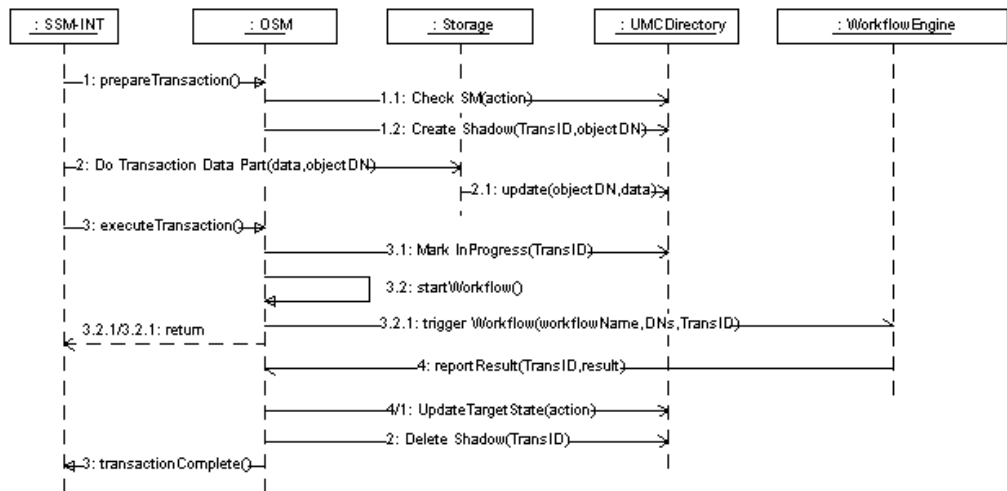
If, for any reason, the workflow is cancelled, the transaction is aborted; that is, the target object is restored to its original state before the beginning of the transaction and the lock is released. The workflow is responsible for rolling back the actions performed because of its execution.

On termination of the transaction the client component is notified. The transaction is then irreversible.

Executing Transactions

As summarized in Figure 63, three messages involving the OSM and its client are employed for every transaction: prepareTransaction, executeTransaction, and transactionComplete. Operations on the target of a transaction can be performed only after prepareTransaction is successfully executed and before executeTransaction is sent. Updating the target object after the executeTransaction will not affect the ongoing transaction. However, this information is not guaranteed to persist, because, if the transaction is aborted, the contents of the target will reflect its state before the prepareTransaction message is executed. The transaction is over only after the client receives the transactionComplete (or transactionAborted in case of abortion) message. Every OSM client must respect this contract, or risk losing data or placing ongoing transactions in an invalid state.

Figure 63: Executing Transactions



Socket Interface

The OSM interfaces with the workflow engine through its native socket interface to send requests and also receives reports from its native interface. Messages are exchanged over a TCP/IP connection.

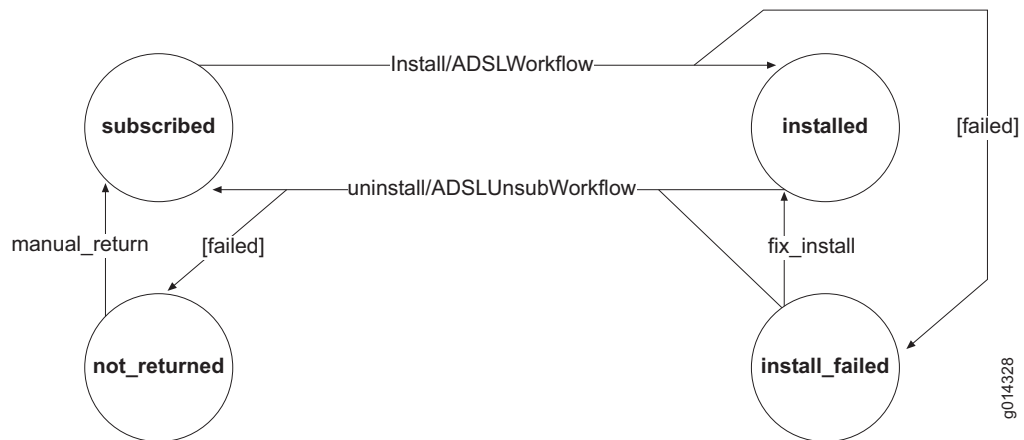
Web Interface

The Web interface for workflows is implemented through the OSM interface for the Web (OSMW). The OSMW is implemented as a Java-based Web application; that is, a servlet. A Web host capable of running servlets is the host of the application.

State Machine

A transactional object state machine is exemplified in Figure 64. The trigger represents an action that the client is attempting to perform on the target. The action through the script name specifies the side effect (workflow) to be executed. The guard (in square brackets in Figure 64) is the completion condition of the workflow; that is, succeeded or failed. The cancellation of a workflow is not represented. When a cancellation occurs, the workflow is rolled back, and the target object moves back to its original state before the transaction. The specification of a completion condition is optional. If not explicitly specified, it defaults to “success.” For example, the transition from “subscribed” to “installed” does not need to specify “success” as the completion condition of the side effect.

Figure 64: State Machine Associated with a Transactional Object



The first action on every object is “create.” If the workflow is cancelled, the object creation is cancelled, causing its deletion. Analogously, the last action is “delete,” which removes the target from the directory. The latter, however, is not mandatory. Depending on the business logic, an object once created cannot be deleted.

If a workflow specified as a side effect of a transition is capable of failing, there must be a transition specifying a completion condition of failure, as in the case of executing “uninstall” from “installed.” If “ADSLUnsubWorkflow” succeeds, it moves to “subscribed.” If the workflow fails the state becomes “not_returned.” If a failure transition is not specified and the side effect workflow fails, the target goes to an exceptional state named “BAD.” It can also reach this special state if the side effect was aborted and was not able to correctly roll back; that is, the workflow experienced a hard cancellation. Once in this state, no actions can be performed on this target, and the intervention of an operator is required to manually change the state back to a valid one. The state enforces that corrective measures be taken before the object returns to normal operation. Such an occurrence denotes a bad design of the state machine or the associated workflow.

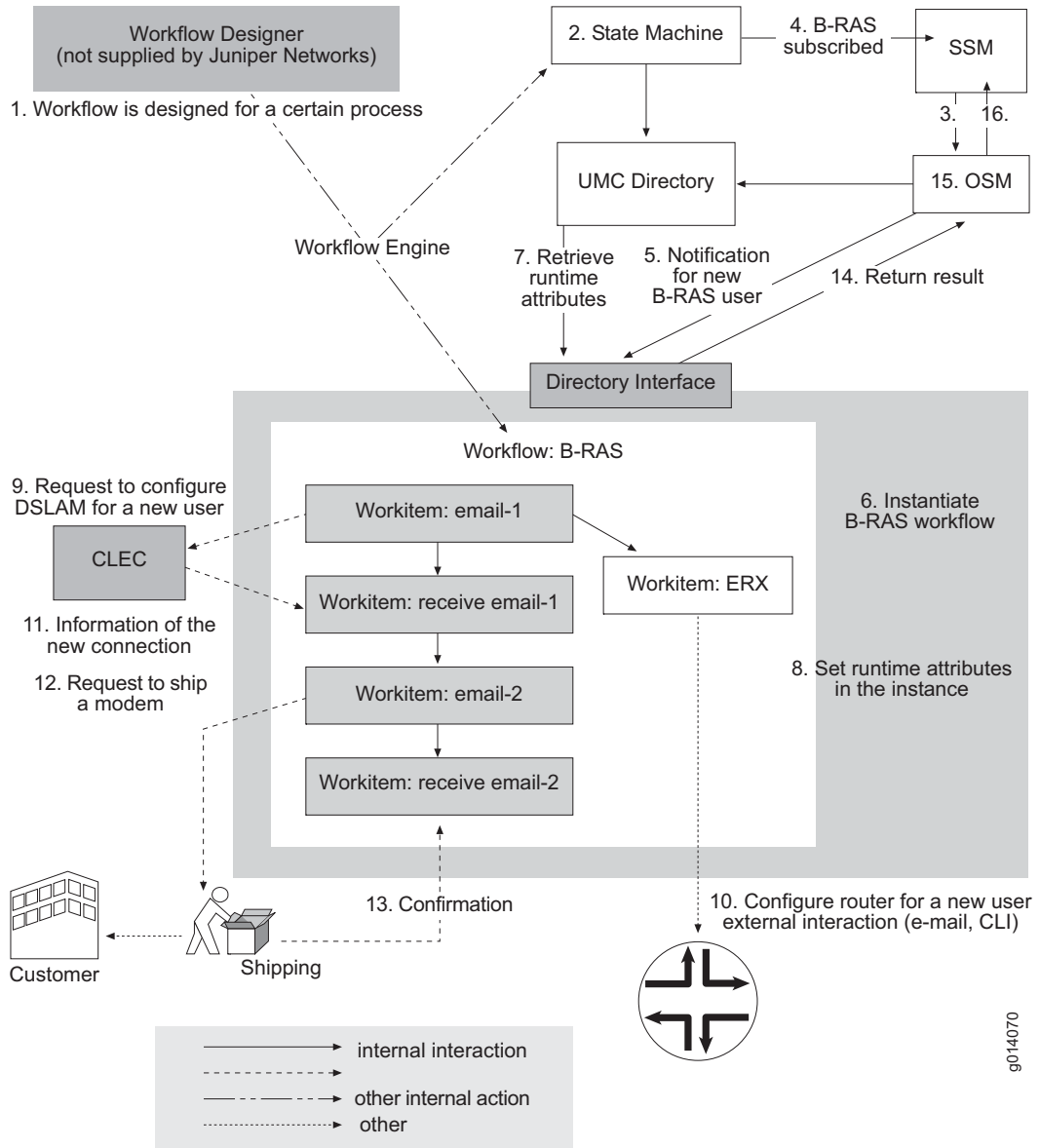
Transaction Execution Example

Figure 65 depicts the transaction execution for B-RAS service provisioning with the Workflow application. The following is the sequence of events:

1. Workflow B-RAS is designed and then is stored in the workflow library.
2. The state machine is designed to manage the life cycle of a service and is stored in the SDX Directory.
3. The transaction is initiated.
4. A new user subscribes to B-RAS service, and a service object is created in the repository (directory).
5. The workflow engine receives a creation notification that will execute a workflow for the B-RAS object for this user.
6. The workflow engine instantiates for workflow B-RAS.
7. The workflow engine retrieves run-time attributes from the repository. A list of run-time attributes is decided during design of the workflow (Step 1).
8. The workflow engine starts the workflow and then waits for the workflow to complete all tasks.
9. The first work item, email-1, sends a request to configure a connection between DSLAM and the router.
10. Parallel to Step 7, workitem ERX sends a request to configure the router.
11. Work item receive email-1 receives information about the new connection.
12. Work item email-2 sends a request to ship an ADSL modem to the user.
13. Work item receive email-2 receives confirmation of the shipment.
14. All tasks are finished. The workflow engine returns a result.

15. The transaction is committed.
16. The result of the transaction is reported to the service and subscriber management (SSM) system.

Figure 65: Example: B-RAS Service Provisioning Lifecycle



Creating a State Machine

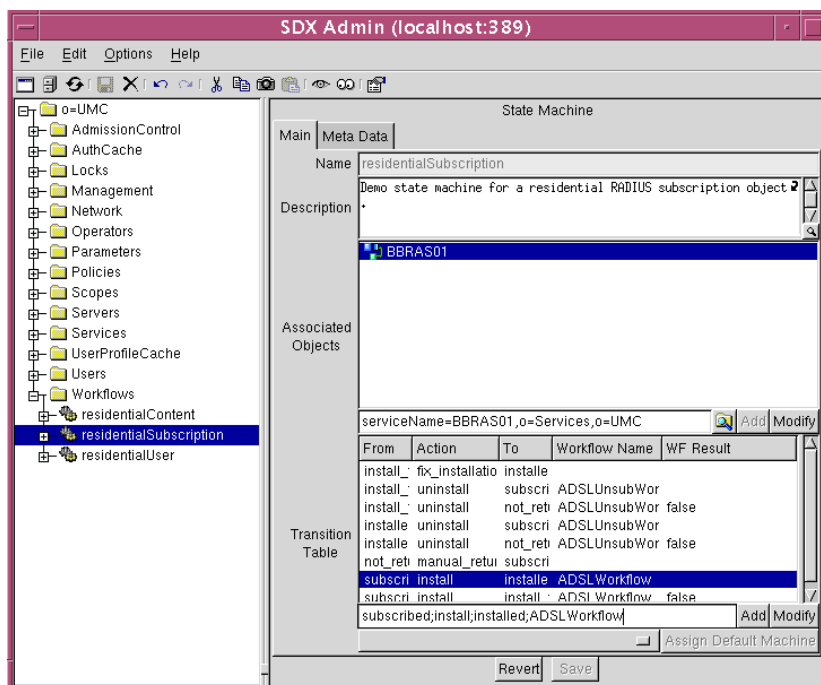
Referring to Figure 66, create the state machine object under Workflows. Then assign the optional associated objects to the state machine. The DNs of the associated objects are passed to the workflow engine when any workflow specified as a side effect of the state machine's transitions is executed.

Next, create transitions of the state machine by adding the transition to the transitions table. For example, the highlighted field in Figure 66 corresponds to the transition from the subscribed state to the installed state in Figure 64.



NOTE: The sample state machine is included in the sample directory.

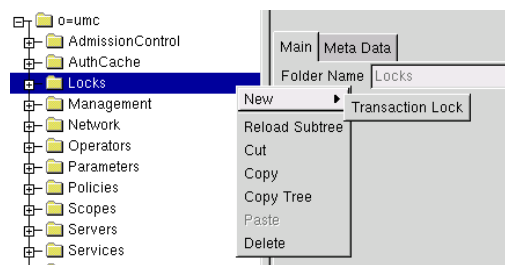
Figure 66: State Machine



Locking a Workflow Transaction

Use this function to lock a workflow transaction. A workflow is locked while it is executing a transaction or when there is a problem with the workflow. This task is performed automatically by the OSM and should not be used unless you are debugging the system. The information present in a lock allows you to trace the state of a transaction and the workflow it is executing (if any), and to abort such a transaction.

1. Highlight Locks, and right-click.
2. Select New > Transaction Lock.



The New Transaction Lock dialog box appears.

3. Enter the lock information, and click OK.

