ServiceControl Manager Tools

DONALD SUIT
HEWLET PACKARD
3404 HARMONY ROAD, MS 99
FORT COLLINS, CO 80528-9599
(970) 898-0327
(970) 898-2838 FAX
dsuit@fc.hp.com

JANUARY 14, 2000

This page intentionally blank

Biographical Sketch

Donald Suit is a software engineer at Hewlett-Packard's UNIX Development Lab in Fort Collins, Colorado. He has over 19 years of software development experience developing various commercial and government applications. He has a B.S. degree in computer science from the University of Maryland and an M.S. degree in computer science from Johns Hopkins University.

Mailing address:

Donald Suit
3404 East Harmony Road, MS 99
Fort Collins, CO 80528-9599

E-mail address:

dsuit@fc.hp.com

1 Introduction

The ServiceControl Manager tool feature provides a user-configurable, role-based mechanism for defining procedures (tools) that system administrators can use to manage from a central management server and direct to execute across a cluster of managed nodes. System administrators can define tools to copy files or execute processes. In the ServiceControl environment, system administrators can define roles, such as database administrator or web administrator, which the system administrators can associate with given tools. The authorization facility of the ServiceControl Management system associates users with roles and nodes within a managed system. With authorizations in place, administrators can execute ServiceControl Manager tools from the central management server, and the tool facility can automatically determine on which managed nodes the tool has authorization to execute and send the tool to execute on those managed nodes.

2 Purpose

The SCM tool feature provides a mechanism to distribute routine or special-purpose tasks across a cluster of HP-UX nodes. The roles and authorizations defined in the SCM, provide a more controlled distributed mechanism than remsh. SCM users can package their favorite scripts in an SCM tool, which the user can distribute and execute across the network of SCM-managed nodes. SCM users can take their collection of favorite scripts and easily package them in an SCM tool. The SCM user can build a collection of SCM toolkits for various purposes, for example trouble-shooting or system performance analysis. SCM role-based authorizations allow the user to assign logical roles to the tools and authorize these roles to the appropriate personnel on any subset of the SCM managed nodes. The authorized personnel can execute their assigned tools to perform the functions associated with their logical roles across the subset of managed nodes. Additionally, HP provides a set of built-in SCM tools for the convenience of SCM users. These tools include Ignite-UX(IUX) and Software Distributor(SD).

3 ServiceControl Manager Overview

A ServiceControl Manager is used to manage a collection of HP-UX nodes that are managed from a single Central Management Server (CMS). The primary purpose of the ServiceControl Manager environment is to increase the efficiency of managing multiple HP-UX systems. The focus of the ServiceControl Manager is the ongoing administration and configuration of HP-UX server systems. The ServiceControl Manager provides the means for administrators to perform tasks on multiple HP-UX nodes in parallel using a secure network protocol. The ServiceControl Manager does not support s700 workstations.

ServiceControl Manager runs on a CMS and from it manages the nodes. The CMS is an HP-UX 11.x server running the ServiceControl Manager software. Because the CMS does not excessively use resources, the CMS server can be used for other purposes in addition to serving as the CMS. It is likely, however, that there will be some specifics regarding the CMS system configuration that will be required (including such things as kernel tunables, file system space, and the configuration of certain daemons).

ServiceControl Manager 1.0 will support a single instance of ServiceControl Manager on a single CMS. All tasks performed on the ServiceControl Manager must be initiated on the CMS although the CMS is accessible via a web interface. The workstation at which a user sits, therefore, just needs web access over a network to the CMS in order to perform tasks. The CMS is the host for the ServiceControl Manager software, the data repository, a web server that allows web access to the ServiceControl Manager, an SD depot containing products used in the configuring of nodes, and an Ignite-UX server.

ServiceControl Manager supports multiple users performing tasks simultaneously (numeric goals and limits are described below under Additional Product Information). There are some limits in how many simultaneous tasks can be performed depending upon the number of target nodes performing the tasks and the disk and memory resources of the CMS. Therefore, users may encounter error messages that indicate that a task cannot be started due to resource constraints.

For more detailed information on ServiceControl Manager, see the paper “Service Control Manager” also in these proceedings.

3.1 Users

ServiceControl Manager uses the HP-UX user authentication mechanism. That means that that for users using the command line to access ServiceControl Manager, it will rely on the authentication done at login and not require any additional passwords. For users accessing ServiceControl Manager via the web, the user's normal HP-UX login name and password will be requested and validated in a way similar to login (1). This decision is a direct result of customer data indicating that the creation of ServiceControl Manager -unique user IDs and passwords is unacceptable. A ServiceControl Manager user is really just an HP-UX user that is now known to the ServiceControl Manager and has some privileges and/or management roles (defined later). In order to start ServiceControl Manager or execute any ServiceControl Manager tools, a user must have been added to the ServiceControl Manager either via the GUI or the CLI. The definition of a ServiceControl Manager user consists of:

- The user's login name,
- A ServiceControl Manager -specific comment , and
- A Trusted User privileges flag.

ServiceControl Manager is able to access the HP-UX user registry (/etc/passwd or NIS/NIS+) to get and display user information, including:

- The user's real name,
- Telephone number, and
- Office location.

A ServiceControl Manager user can, depending upon the roles assigned him, manage systems in the ServiceControl Manager via the CMS. In addition, one can examine the ServiceControl Manager log, and scan the node group and role configurations. Another type of ServiceControl Manager user capability is required so that the ServiceControl Manager itself can be managed. This other capability is called the ServiceControl Manager Trusted User privilege.

A ServiceControl Manager Trusted User is a ServiceControl Manager user responsible for the configuration and general administration of the ServiceControl Manager. The capabilities restricted to this type of user are:

Create/Modify User Security Profile

Add/Modify/Delete a Node

Add/Modify/Delete a Node Group

Tool Modification

Tool Authorization

The granting of these privileges implies a trust that the ServiceControl Manager user will responsibly configure and maintain the overall structure of the ServiceControl Manager.

3.2 Managed Nodes

In a ServiceControl Manager, the managed HP-UX servers are referred to as “managed nodes” or simply as “nodes”. The concept of a node is that it represents a single instance of HP-UX running on some hardware. ServiceControl Manager is designed to manage HP-UX system instances in any form.

The SCM Trusted User must explicitly add nodes other than the CMS to the ServiceControl Manager cluster. The process of adding a node to a ServiceControl Manager includes a minimum of installing some SD products from the CMS onto the node and performing an “add node” operation in ServiceControl Manager.

ServiceControl Manager nodes run a managed node agent that performs the management tasks sent to them from ServiceControl Manager. In addition, there will likely be some client side software installed on each node associated with the various management applications supported in ServiceControl Manager. For example, this might include the SD agent and the DMI agent. These are all items that are included when SD is used to install the ServiceControl Manager node product on a new node.

The user interface runs only on the CMS, although its display can be accessed over the network via the web or X windows. No other node in the ServiceControl Manager is capable of executing remote tasks, accessing the repository, or any other ServiceControl Manager operations. The nodes are capable of contacting the CMS when the agent process is started up to notify the ServiceControl Manager process that it is accessible, but are not able to initiate actions on the CMS.

The following steps occur during the removal of a node from the CMS:

- remove the node from the repository
- remove the node from any node groups of which it is a member
- remove any authorizations for the node

No software, ServiceControl Manager-related or otherwise, is removed from the managed node. Rather, the node’s CMS context, i.e. its presence in the repository, node groups and authorizations, is deleted.

3.3 Node Groups

ServiceControl Manager node groups are collections of nodes in a ServiceControl Manager. They can have overlapping memberships, such that a single node can be a member of more than one group. An anticipated use of ServiceControl Manager node groups is as a convenient way to specify multiple targets for distributed tasks and multi-computer aware applications. The grouping mechanism allows flexible partitioning of the ServiceControl Manager so those customers can use it to reflect the way nodes are already grouped in their environment (whether or not they are currently grouped explicitly).

A node group definition includes of a group name, a list of members, and a description.

Any ServiceControl Manager user (defined below) may view node groups. Configuration of node groups is a restricted operation available only to ServiceControl Manager Trusted Users (also defined below).

3.4 Roles

The ServiceControl Manager role defines what the user is able to do on the associated node or node group. Therefore a ServiceControl Manager role is essentially just a group of tools. Each role can have one or more tools and each tool can belong to one or more roles. When users are given the authority to perform some limited set of functionality on one or more nodes, the authorization is done based upon roles and not on tools. The ServiceControl Manager role allows

the sum total of functionality represented by all the tools to be divided into logical sets that correspond to the responsibilities that would be given to the various administrators.

For example, one ServiceControl Manager role might be that of performing backups. The “backup” role would contain tools that perform backups, manage scheduled backups, view backup status, etc. In this example, a user could be assigned the “backup” role for a group of systems that run a specific application. Later, when new backup tools are created and then added to the “backup” role, this user is immediately given access to the new tools on those systems.

ServiceControl Manager 1.0 supports a fixed number of roles currently set at 16. There is one fixed role, for now called “Master Role”, but the names of the other ServiceControl Manager roles can be modified. “Master Role” is the default role assignment for all newly created tools. In fact, all tools are assigned to “Master Role” and cannot be removed from it. The notion is that this role is analogous to someone who actually knows the root password on a node and hence can do anything they want. This means that users that have the “Master Role” role on one or more nodes can run any tool on those nodes.

The other 15 roles can be renamed as needed by a Trusted User. They can also be disabled. This allows a user who only has access to the systems periodically, such as an auditor or an HP field engineer, to be granted authorizations via a role that is only enabled when needed.

3.5 Authorizations

The ServiceControl Manager authorization model supports the notion of assigning to users the ability to run some set of tools on some set of nodes. The authorization object is an association that links a user to a role on either a node or a group. When users are given the authority to perform some limited set of functionality on one or more nodes, the authorization is done based upon roles and not on tools. The role allows the sum total of functionality represented by all the tools to be divided into logical sets that correspond to the responsibilities that would be given to the various administrators.

Each authorization links a single user to a single role and to a single node. Each role corresponds to one or more tools and each tool can be associated with one or more roles. Each user can be assigned multiple authorizations, as can each role and each node.

The ServiceControl Manager authorization model for determining if a user can execute a tool on a set of nodes is defined by an “all or none” model. Therefore, the user must have a valid authentication association for each target node to execute the tool. If even one authorization does not exist for one of the nodes, the tool execution fails.

3.6 Tools

A central part of ServiceControl Manager is the ability to execute various management commands or applications on the one or more nodes. The commands or applications must be wrapped in a ServiceControl Manager tool. Users can define ServiceControl Manager tools that run simple commands like bdf (1) or mount (1M), or launch single system interactive applications such as SAM or EMS 3.0.

Additionally, users can define tools to copy files from the CMS to the target nodes. This provides the user the ability to push out copies of configuration files that need to be kept consistent across a specific set of nodes. It is not intended as a replacement for the Software Distributor and is limited in the number (16 files) and type of files (only regular files - no device files, symlinks, etc.) copied. Files can only be copied from the CMS to the managed nodes - not from the nodes back to the CMS.

The combination of being able to copy a small number of files to a target node and then to execute a specific command allows for tools to be created that copy a script to a target node and then execute it once it is there. Another way to use the combination would be to copy one or more data files to a node (such as a crontab file) and then execute a command that uses the file

contents. When files are copied from the CMS to a managed node, the ownership and permissions of the file on the CMS are copied and applied to the copy placed on the managed node. If the file in question already exists on the managed node in the destination location, that file is first removed. If a directory exists on the managed node in the destination location, the file copy and the rest of the task fails.

The execution command string is not required for tools that specify one or more files to be copied. This is so that a tool can be created that merely distributes files and does no other operations.

3.7 Log

An integral part of the ServiceControl Manager functionality is the ability to record and maintain a history of events. Customers have expressed the need to track which users have initiated which events, with details such as launch time, target nodes, and results. In response to this need, both ServiceControl Manager configuration changes and task execution events will be logged.

ServiceControl Manager configuration changes include adding, modifying and deleting users, nodes, node groups, tools, authorizations or modifying roles in the ServiceControl Manager.

Task execution events include the details and intermediate events associated with the running of a tool. The details include the identity of the user who launched the task, the task identifier, the task start time, the actual tool and command line with arguments, and the list of target nodes. The intermediate events that are logged include the beginning of a task on a managed node, any exceptions that occur in attempting to run a tool on a node, and the final result (if any) of the task. The exit code, stdout and stderr (if they exist) will also be logged.

3.8 Repository

ServiceControl Manager requires a persistent centralized data repository for general management information. The ServiceControl Manager data repository uses the Lightweight Directory Access Protocol (LDAP) and the Netscape Directory Service (NDS) as its backing store.

All definitions of tools, users, nodes, groups, roles, and authorizations are stored in the SCM repository.

4 ServiceControl Manager Tools In Depth

4.1 Tool Attributes

4.1.1 Name

The tool name is a unique identifier used to address the tool for addition, modification, removal, listing, or execution.

4.1.2 Category

The tool category is a name with which the operator relates this tool with other tools. This allows an operator to logically associate tools to aid in managing the tools.

4.1.3 Description

The description is a short one-line description of the tool.

4.1.4 Owner

The owner attribute defines the ServiceControl Manager user that owns the tool. The value of the owner attribute determines who may modify the tool and the roles that are enabled for the tool.

If the owner is specified, only the “Master Role” role is valid, all other roles that may have been assigned are temporarily disabled. The ServiceControl Manager user identified by the owner attribute can modify the tool but can not modify roles or change the owner field.

If the owner field is empty, all roles assigned to this tool are enabled, and only a ServiceControl Manager user with “Trusted User” privileges can modify the tool.

4.1.5 Comment

The comment may be a multi-line text field to provide instructions, comments, warnings, etc.

4.1.6 Command

The command attribute represents the invariant portion of the command line that the SCM will execute as the tool. If the user provides no parameters to the tool definition, the command represents the entire command line of the tool. If the user provides parameters to the tool definition, the SCM concatenates the parameters and argument values to the command to construct the tool’s complete command line.

4.1.7 Parameters

Parameters are a list of argument entries. These are composed of an optional prefix, an optional prompt, and a flag indicating whether the argument is optional or required. The user must define either the prefix or the prompt to create a valid parameter.

The prefix is an invariant string that the ServiceControl Manager will concatenate to the command string as the ServiceControl Manager is building the command string for execution.

The prompt is an informational string used by the ServiceControl Manager GUI to describe a value to be concatenated to the command string.

When building the command line to be executed, the ServiceControl Manager iterates through the list of parameters, if specified. If a parameter is required, the ServiceControl Manager looks for the prefix and, if it is present, appends the prefix to the command line. If a prompt is present, then the ServiceControl Manager looks for an operator-provided argument value. The ServiceControl Manager appends the argument value to the command line. If a parameter is not required, the ServiceControl Manager only concatenates the parameter if the operator provides a value for the argument. Because of this, the ServiceControl Manager does not allow parameters specifications that are optional and do not provide a prompt with or without a prefix.

4.1.8 File–copy pairs

File–copy pairs are a list of files to be copied. These are specified as a pairing of a source pathname on the CMS and a destination pathname on the managed node. The ServiceControl Manager does not allow more than one file–copy pair with the same destination in one tool.

4.1.9 Execution user

When the tool is executed on a managed node, the tool will run under this user’s UID.

4.1.10 Roles

The roles attribute provides the list of ServiceControl Manager roles for this tool. Whether or not these roles are enabled for this tool depends on the value of the owner attribute. If the owner

attribute is not specified, then the roles are enabled. If the owner attribute is a ServiceControl Manager user, then only the “Master Role” role is enabled. In order for a ServiceControl Manager user to run the tool on a managed node, the ServiceControl Manager user must be authorized with one of the tool’s enabled roles on the target managed node.

4.1.11 Default Targets

The default targets attribute specifies the possible managed nodes on which the tool may run if the operator does not specify a target manage node. The possible values are:

- none specified, i.e. no defaults,
- “ALL”, all managed nodes for which the operator is authorized,
- “CMS”, only run on the managed node specified as the CMS, or
- A single node on which the user must be authorized.

4.1.12 Log flag

The log flag indicates where to save the stdout and stderr results from the execution of the tool to the log. By default the results are saved to the log.

4.1.13 Launch only flag

The Launch only flag indicates whether the agent should wait for the completion of the command and capture the exit code, stdout, and stderr results. If set, the agent does not wait for the completion of the command. By default, the agent does wait for the command to complete and captures the results to be returned to the CMS.

4.2 Managing Tools

The ServiceControl Manager provides two mechanisms for managing tools, a graphical user interface (GUI) and a command line interface (CLI). Both interfaces provide the same basic capabilities, to define, add, modify, remove, and execute tools. The ServiceControl Manager GUI provides a windowing look-and-feel for managing tools. The GUI provides screens for defining tools and menu items for adding, modifying, removing, and executing tools. The GUI is more interactive and dynamic than the CLI. The CLI provides access to tools at a terminal prompt and allows tool management in a shell script.

4.2.1 Defining Tools

The ServiceControl Manager provides mechanisms for defining tools. The SCM restricts who may specify certain attributes of a tool, such as the owner or tool roles. The SCM only allows a trusted user to specify the owner of a tool or remove the owner field from a tool. If an SCM non-trusted user is defining a tool, the SCM will automatically set the tool owner to the non-trusted user’s id. A non-trusted user may only specify the SCM “Master Role” in a tool. The “Master Role” is the default role for all SCM tools, so assigning this role has no real effect on the tool definition. The SCM restricts assignment of these fields because the values of these fields affect the execution of the tools on the target-managed nodes. If the owner field is set, then only the “Master Role” is enabled in the tool; therefore, the user can only execute the tool on managed nodes for which the SCM has authorized the user with the “Master Role” role. If the owner field is not set, the SCM enables all roles assigned to a tool. Any user with authorizations matching those in the tool to the appropriate managed nodes, may execute the tool on those nodes.

4.2.1.1 Graphical User Interface

Show GUI windows that allow specifying tools.

4.2.1.2 Tool Definition File

For defining tools to be added with the command line interface, SCM provides a mechanism for translating a tool definition file into an SCM tool. The tool definition file may contain one or more tool definitions. The SCM tool definition grammar description uses the EBNF syntax, where "|" means a choice, "?" optional, and "*" means zero or more. The tool definition grammar is as follows:

```
file           := toolDefinition*

toolDefinition := localTool

localTool      := "SSA" "tool" alphaString "{" localToolGuts "}"

localToolGuts  := toolGlobals* copyStmt? localExecStmt? defTargets?
roleList?

toolGlobals    := "description" string |
                  "comment" string     |
                  "revision" string    |
                  "owner" name         |
                  "category" alphaString

copyStmt       := "copy" "{" fileCopySpec ( "," fileCopySpec )* "}"

fileCopySpec   := filePath ":" filePath

filePath       := "/" name ( "/" name )*

localExecStmt  := "execute" "{" commandSpec toolOpts* "}"

commandSpec    := "command" cmdString ( argStmt )*

argStmt        := "arguments" "{" promptStmt ( "," promptStmt )* "}"

promptStmt     := cmdString? ( ":" alphaString ( "optional" )? )?

toolOpts       := ( launchOnly | logOutput | executeUser )

launchOnly     := ( "launch" | "nolaunch" )

logOutput      := ( "log" | "nolog" )

executeUser    := "user" name

defTargets     := "default" "targets" ( "CMS" | "all" | name )

roleList       := "roles" "{" alphaString ( "," alphaString )* "}"
```

When a user sets logOutput to “log”, the SCM logs output from the execution of the tool to the Central Management Server (CMS) log file /var/opt/mx/logs/scmgr.log.

Example tool definition:

```
SSA tool “Sample Tool” {
    description “This is a one line description of a sample tool”
    comment “This is a longer description to provide more information,
             perhaps even usage information”
    revision “1.0”
    owner dsuit
    category “Don’s Tools”
```

```
    copy {
        /var/etc/sourcefile : /var/etc/destinationfile,
        /var/etc/source2    : /var/etc/dest2
    }
    execute {
        command “ls “
        arguments {
            “-l “: ,
            :” Enter path: “ optional
        }
        nolaunch
        log
        user root
    }
    default targets “ALL”
    roles {
        operator,
        “Master Role”
    }

}
```

The above definition defines a tool named “Sample Tool” that is owned by user dsuit. The SCM will put this tool in the “Don’s Tools” category. The tool will copy two files from the CMS to the target nodes. After copying the two files, the tool will execute the ls command with the –l option, and, optionally, the user may specify the file path for the ls command. When this tool is executed, the SCM will wait until the tool completes on all nodes and provide the output directly to the user. The SCM will write the tool execution output to the SCM log. The SCM will run the tool as the “root” user on the target nodes. If the user does not provide a list of target nodes, the tool will run on all nodes for which the user is authorized. To run the tool, the user must have either the operator or “Master Role” role on each of the target nodes.

4.2.2 Adding tools

Once the user has defined a tool, the user may add the tool to the SCM repository. Assuming the user defines the tool properly and the tool does not currently exist in the SCM repository, the SCM will allow the user to add the tool.

4.2.2.1 Graphical User Interface

Show the GUI window that adds a tool.

4.2.2.2 Command Line Interface

To add tools from the command line interface, the user must specify the add option and provide a tool definition file as a parameter to an mxtool command. For example to add tools from the mytools tool definition file, the user would enter the following command line:

```
mxtool –a –f mytools
```

When the user enters the mxtool command, the SCM parses the tool definition file and adds the resulting tools to the SCM tool repository. If SCM encounters any errors during tool definition parsing, the SCM will report the errors and will not add any tools to its repository. Before adding the parsed tool to its repository, the SCM will check if a tool by the same name exists. If a tool with the same name exists, the SCM will not add the parsed tool, but the SCM will continue to try to add the remaining parsed tools.

4.2.3 Modifying tools

Users may modify tools that exist in the SCM repository under the proper circumstances. SCM trusted users may modify any tool in the SCM repository. For a non-trusted user to modify an SCM tool, a trusted user must set the owner field to the non-trusted user's id. This action will disable the roles of the tool, except for the "Master Role", and the SCM will not permit the non-trusted user to modify the owner attribute or the tool's roles.

4.2.3.1 Graphical User Interface

Show the GUI window for modifying tools.

4.2.3.2 Command Line Interface

To modify tools from the command line interface, the user must specify the modify option and provide a tool definition file as a parameter to an mxtool command. For example to modify tools from the mytools tool definition file, the user would enter the following command line:

```
mxtool –m –f mytools
```

When the user enters the mxtool command, the SCM parses the tool definition file and adds the resulting tools to the SCM tool repository. If SCM encounters any errors during tool definition parsing, the SCM will report the errors and will not modify any tools to its repository. Before the SCM modifies the tool based on the parsed tool, the SCM must try to read the existing tool with the same name from its repository. If the SCM cannot find a tool by the same name, the SCM will skip the parsed tool, but the SCM will try to modify the remaining tools that were parsed from the given tool definition file.

4.2.4 Removing tools

Only trusted users may remove tools from the SCM repository.

4.2.4.1 Graphical User Interface

Show the GUI window to remove tools.

4.2.4.2 Command Line Interface

To remove tools from the command line interface, the user must specify the remove option and provide a list of one or more tool names as a parameter to an mxtool command. For example to remove the tool named "Sample Tool" from the SCM tool repository, the user would enter the following command line:

```
mxtool –r –t "Sample Tool"
```

The user may enter one or more tool names on the command line. The SCM will attempt to remove the tools provided in the command line. If the SCM does not find a specified tool, the SCM will proceed to attempt to remove the next specified tool.

4.2.5 Listing tool information

The SCM provides mechanisms for displaying the tools, the tool categories, and the tool definitions to the SCM users. Any SCM user may examine tool information.

4.2.5.1 Graphical User Interface

Show the GUI windows that display tool information.

4.2.5.2 Command Line Interface

To display tool information using the command line interface, the user must specify the list option, a list option format argument, and may optionally provide either a tool category or a list of one or more tool names as a parameter to an mxtool command. If the user does not provide a category or list of tool names, the SCM will list the entire SCM tool repository in the specified format.

The mxtool CLI provides four list option arguments to control the listing format of a tool:

- n – lists just the tool names
- t – lists the tool category, tool names, and description in a row, column format
- d – lists all of the tool's attribute values in a single column, screen-viewable format
- f – lists the tools in a tool definition file format

For example, the following command lists the tool, "Sample Tool", in a tool definition format and directs the output to a file named "mySampleTool":

```
mxtool –lf –t "Sample Tool" > mySampleTool
```

4.3 Executing Tools

4.3.1 Generating the executable command

The SCM provides mechanisms to execute tool from the GUI or the CLI. To execute a tool, the user must at a minimum provide the tool name. If the tool does not specify a default target in the tool definition, the SCM user must specify the managed node or nodes on which the tool may run. Additionally, if the tool definition requires additional parameter values to complete the command line, the user must provide those parameter values. If the tool executes a command line, the SCM will try to construct a command line from the tool definition and the user-provided data. The SCM takes the tool's command attribute and goes through each of the parameters in the tool's parameters list and concatenates the parameter's prefix (if defined) and any provided parameter value.

Users can define SCM tool parameters with a prefix, a prompt, or both. If the user defines only a prefix for the parameter, the user must also define the parameter as required. For parameters that are required, the SCM will generate an error if the tool definition contains a prompt, but the user does not provide a value. For parameters that are optional, the SCM only concatenates the parameter with its optional prefix if the user provides a value. For this reason, the SCM will flag as an error any optional parameters that have no prompt defined.

Once the SCM has concatenated the command and parameters into the complete command line, the SCM will determine the target managed nodes on which the tool will run.

4.3.1.1 Run A Tool GUI option

4.3.1.2 mxexec CLI command

To execute a tool from the command line, the SCM provides the mxexec command. When entering the mxexec command, the user must at a minimum provide a tool name for example:

mxexec –t "My Tool Name"

If the tool's parameters require argument values or the user wishes to provide optional argument values, the user must provide the desired values, for example:

mxexec –t “My Tool Name” –A value1 “value 2” “value 3”

If the tool does not specify default targets or the user wishes to specify the target nodes, the user must enter the nodes on which the tool will run, for example:

mxexec –t “My Tool Name” –n target1 target2 target 3

The user may specify that the output from the tool execution goes to a single file, for example:

mxexec –t “My Tool Name” –O outputfilepath

The user may specify that the output from the tool execution goes to a directory with a separate output file for each target node, for example:

mxexec –t “My Tool Name” –o outputDirectoryPath

4.3.2 Determining the authorized nodes

Before executing a tool, the SCM will determine if the SCM user is authorized to run on the target managed nodes. The user may specify the target nodes on which the SCM will run the tool, or the user may defer to the nodes defined by the tool’s default targets attribute. When the user provides the list of targets, that list of targets always overrides the tool’s default targets.

When the user does not specify a list of target nodes, the SCM will determine the target nodes from the tool’s default target attribute. If the tool’s default target is “CMS”, the SCM will run the tool on the node that the SCM designates as the CMS server. If the default target is set to a managed node, the SCM will run the tool on the specified managed node. If the tool’s default target attribute is set to “ALL” and the user does not provide a list of target nodes, the SCM will automatically generate the list of authorized nodes by intersecting the tool’s roles list and the user’s SCM authorizations. If the tool definition does not specify a default target, the user must provide the target nodes on which the tool must run.

Once the SCM determines the target nodes, the SCM will check if the tool’s roles and the user’s SCM authorizations intersect for the provided target nodes. The user’s SCM authorization consists of the user’s name, a role, and a managed node. The user may have many authorizations to get the proper set of roles associated with the appropriate nodes. The SCM will only allow a user to run a tool for which the user has the one of the tool’s roles authorized for the target managed node.

4.3.3 Running the executable command

When a SCM user runs a tool, the result is a SCM task. The SCM Distributed Task Facility (DTF) manages SCM tasks. The DTF consists of two processes, a DTF daemon, which runs on the CMS, and DTF agents, which run on the SCM-managed nodes. A task represents the invocation of a specific tool on a specific set of target nodes at a specific time. Therefore a task has a lifecycle and a lifetime. The lifecycle of a task is the sequence of states that a task steps through from definition to completion. The lifetime of a task is the time duration from when a task is started to when it completes. After defining a task and before handing off the task to the DTF agents on the target managed nodes, the DTF daemon assigns a task identifier for each task. The SCM user can use this identifier to track the task and to look up information later about the task in the SCM log.

For every target node, a task goes through a set of states that track the progress of the task on each node. The states are:

- Pending,
- Contacting target,

- Copying files,
- Running tool, and
- Complete.

The following table describes each of the task states:

State	Description
Pending	Nothing regarding the specific target is happening. This state is used when there is a very large number of target-managed nodes and the DTF daemon is only able to run a task in parallel on a smaller number of nodes. Those targets that are waiting to start are in the Pending state.
Contacting target	During this state, the DTF daemon makes contact with the DTF agent on the target node, establishes a connection, and passes the task information to the DTF agent.
Copying files	If there are any files to copy, the task enters this state, and the DTF daemon transmits the contents of the files to the target managed node. The target's DTF agent writes the files and sets the files' ownership and permissions.
Running tool	If there is a command line to execute (the command line is optional for a tool), the DTF daemon assigns the Running tool state to the task. During this state, the target's DTF agent forks a process to run the command, establishes a clean process environment, and executes the POSIX shell with the command line as the argument. The DTF agent sets the stdin for the process to /dev/null. If the tool is a launch-only tool, as soon as the POSIX shell has successfully executed the command line, the target's DTF agent ignores the child process and moves to the next state. If not, the target's DTF agent waits while the command executes and after it exits, it gathers up the stdout, stderr, and exit code of the process, returns the results to the DTF daemon on the CMS, and closes the connection.
Complete	The task has completed, and the DTF daemon on the CMS makes the resultant information available to the user interface and logs the information to the SCM log. The information about a running task is stored dynamically by the DTF daemon. The DTF daemon remains in contact with the nodes on which tools are running and provides status information back to the user interface. As long as a task is running and for a short time after it completes, the DTF daemon retains all the state information about the task, including all of the output and exit codes from each target node. The information about a task is kept in the SCM after the task completes as long as there is a user interface process that has the task open (via the View Task Details screen, for example). Because the logging of tool output is optional (controlled by the tool definition), the log may or may not contain the tool output (stdout and stderr). But the exit code from each target node, if the command was successfully executed, and the failure information, if it was not, is always logged. The user can view the information stored in the log at any time in the future until the log information expires.

When a task is executing on a node, the agent will set the following environment variables in the environment in which the tool command runs:

- MX_USER - This contains the UID of the SCM user who ran the tool.
- MX_TASKID - This is the task id SCM assigned to this task.
- MX_TOOL - This is the SCM tool name (may not be the same as the tool script name).

- MX_TARGETS - This contains a space-separated list of target nodes for multi-system aware applications. This environment variable is empty for applications that are not multi-system aware. The target nodes will be sorted in a lexicographic order.

- MX_CMS - This is the SCM CMS hostname.

- MX_REPOSITORY - This is the hostname of the system containing the NDS repository.

In addition to these special environment variables, the agent also sets some other standard variables:

- DISPLAY - This is set depending upon the way in which the user invoked the SCM. If the tool is invoked by the mxexec command line, the value of DISPLAY in the current environment is used. If the GUI was invoked in a browser and used to run the tool, the SCM creates an X server that provides an X window onto which X applications, including Java windowing applications, can map their windows.

- HOME - This is set depending upon the UID used to invoke the command. The home directory value is looked up from the user registry (/etc/passwd or NIS) and is used to set HOME.

4.3.4 Canceling and killing tool execution

The SCM provides two mechanisms to interrupt task execution, canceling or killing the task. While the task is in a pending or copying files state, the user may cancel the task. Canceling the task interrupts file copying and prevents the DTF daemon from transitioning the task to the running state. Once the task enters the user is limited to killing the task. When the user requests to kill a task, the DTF daemon sends messages to all daemons involved in the task to kill the processes related to the task. Naturally, if a DTF agent receives the kill command near the completion of the task's process, the process may complete before the agent can kill the task.

4.3.5 Examining the results

There are three possible mechanisms to view the results of tool execution:

If the tool was not a launch-only tool, the DTF daemon will send the tool's task output directly to the user. If the tool is run from the GUI, the output goes to an output window. If the tool is run from the CLI, the output goes to standard output and error.

If the DTF daemon is still managing the tool's task, the task output is available by providing the task's id to either the GUI or the CLI's mxexec command.

If the tool definition set the tool's log flag, the DTF daemon will write the tool's task output to the SCM log, and the user may view the SCM log data until the SCM log expires, i.e. is overwritten.