StreamServe Persuasion SP4
Command line utilities

User Guide

Rev A
Contents

Command line utilities ........................................................................................................5
  Using the command line utilities ..................................................................................7
  Arguments for ss_scm utility .....................................................................................8
  Arguments for ss_territory utility .............................................................................12
  Arguments for ss_deploy utility ..............................................................................18
Examples ..........................................................................................................................20
  Creating an application domain ..............................................................................20
  Modifying an existing application domain .............................................................20
  Creating a StreamServer application .....................................................................20
  Creating a service for a StreamServer application ...............................................21
  Deploying an export package to an application ......................................................21
  Modifying a StreamServer service ........................................................................21
  Starting a StreamServer service ............................................................................22
  Checking if a StreamServer service is running ......................................................22
  Stopping a StreamServer service ...........................................................................22
  Creating a web portal and assign it to a domain ......................................................22
  Creating an Archiver node .....................................................................................23
  Creating an Archiver application ............................................................................23
Troubleshooting..............................................................................................................24
  The utilities does not start .....................................................................................24
  Prompt displayed for realm user name ....................................................................24
  HTTP listener error displayed .................................................................................24
Command line utilities

You can use the command line utilities instead of Control Center for example if:

- You cannot connect to the server environment from a computer running Microsoft Windows.
- You want to automate or script certain tasks.
- You need to administer your server environment from within a UNIX environment

Prerequisites
The following must be installed:

- StreamServer Enterprise Repository
- Framework

About the utilities
The following utilities are covered in this documentation:

**ss_territory**
Create and administer application domains and StreamServe applications (Service Gateway, StreamStudio and StreamServer)

**ss_scm**
Create and administer StreamServer and service gateway services.

It is not enough to create applications for StreamServers and service gateways. You must also create local services for these applications.

**Note:** You do not need to create a local service for StreamStudio.

**ss_deploy**
Deploy a Design Center export file to a StreamServer application.

Scenario 1 – Create application domain and run service

1. Use ss_territory to create an application domain.
2. Use ss_territory to create a StreamServer application in the application domain.
3. Use ss_scm to create a StreamServer service.
4. Use ss_deploy to deploy an export package.
5. Use ss_scm to start the StreamServer service.
Scenario 2 – Redeploy a StreamServer service

1. Use ss_scm to stop the StreamServer service
2. Use ss_deploy to deploy an export package.
3. Use ss_scm to start the StreamServer service.
Using the command line utilities

UNIX

1. Set the $STRS_LOCATION environment variable to where you installed the StreamServe software, for example:

   STRS_LOCATION=/usr/streamserve/streamserve-5.3.0.GA.123
   export STRS_LOCATION

2. Browse to the following directory:

   $STRS_LOCATION/applications/managementgateway/bin

3. Run the following command:

   ./start <utility> <arguments>

   For ss_scm arguments, see Arguments for ss_scm utility on page 8.
   For ss_territory arguments, see Arguments for ss_territory utility on page 12.
   For ss_deploy arguments, see Arguments for ss_deploy utility on page 18.
   For examples, see Examples on page 20.

Windows

1. In the command line window, browse to the following directory:

   <StreamServe_installation>\Platform\Core\1.2\bin

2. Run the following command:

   <utility>.exe <arguments>

   For ss_scm arguments, see Arguments for ss_scm utility on page 8.
   For ss_territory arguments, see Arguments for ss_territory utility on page 12.
   For ss_deploy arguments, see Arguments for ss_deploy utility on page 18.
   For examples, see Examples on page 20.
Arguments for *ss_scm* utility

**Required arguments**

- `-servicename <name>` where `<name>` is the service name
- `-action <action>` where `<action>` is one of the actions in the table below.

**Note:** The `-servicename <name>` argument is not required if `-action <action>` argument is `-action uninstall` or `-action printservices`

<table>
<thead>
<tr>
<th><code>&lt;action&gt;</code></th>
<th>Description</th>
<th>Requires additional arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>Starts a service.</td>
<td></td>
</tr>
<tr>
<td>stop</td>
<td>Stops a service.</td>
<td></td>
</tr>
<tr>
<td>isrunning</td>
<td>Checks whether a service is running or not.</td>
<td></td>
</tr>
<tr>
<td>pause</td>
<td>Pauses a service</td>
<td></td>
</tr>
<tr>
<td>resume</td>
<td>Resumes a service</td>
<td></td>
</tr>
<tr>
<td>newname</td>
<td>Renames a service.</td>
<td><code>-newname &lt;name&gt;</code></td>
</tr>
<tr>
<td>setstartuptype</td>
<td>Changes the startup type of the services.</td>
<td><code>-startuptype &lt;startuptype&gt;</code></td>
</tr>
<tr>
<td>setstartupuid</td>
<td>Changes the server login username.</td>
<td><code>-uid &lt;username&gt;</code></td>
</tr>
<tr>
<td>setservicetype</td>
<td>Sets the service type.</td>
<td><code>-servicetype &lt;type&gt;</code></td>
</tr>
<tr>
<td>setdescription</td>
<td>Sets the service description.</td>
<td><code>-description &lt;description&gt;</code></td>
</tr>
<tr>
<td>create</td>
<td>Creates a new service instance.</td>
<td><code>-binpath &lt;path&gt;</code> <code>-description &lt;description&gt;</code> <code>-servicetype &lt;servicetype&gt;</code> <code>-startuptype &lt;startuptype&gt;</code></td>
</tr>
<tr>
<td>setbinpath</td>
<td>Sets path to the service executable.</td>
<td><code>-binpath &lt;path&gt;</code></td>
</tr>
<tr>
<td>getbinpath</td>
<td>Gets path to the service executable.</td>
<td></td>
</tr>
<tr>
<td><strong>&lt;action&gt;</strong></td>
<td><strong>Description</strong></td>
<td><strong>Requires additional arguments</strong></td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>----------------------------------</td>
</tr>
</tbody>
</table>
| delete     | Deletes the service.  
**Note:** This can corrupt the system if important services are removed | |
| setarg     | Sets a startup argument  
**Note:** The argument must exist. | -argname <argname>  
-argvalue <argvalue> |
| newarg     | Creates a new startup argument for specified service. | -argname <argname>  
-argvalue <argvalue> |
| delsinglearg | Delete a single startup argument e.g. -demo. (A single argument is an argument without a value) | -argname <argname> |
| delbinaryarg | Deletes a binary argument e.g.  
-wd /home/user/projects/myproject/wd | -argname <argname> |
| getstartuptype | Gets the service startup type. | |
| getstartupuid | Gets the service startup uid/username. | |
| getservicetype | Gets the service type. | |
| getdescription | Gets the service description. | |
| printservices | Gets installed services. If -servicetype <servicetype> is specified, only services of that type are returned. | |
| printarguments | Prints startup argument values for the service. | -argname <argname> |
| getarg | Gets startup argument value. | -argname <argname> |
| uninstall | Uninstalls all applications of the specified type. Also unregisters the applications from the application domain. | -applicationtype <type>  
-applicationversion <version> (Optional if action is uninstall) |
| updateappbinpath | Updates the executable path of all applications to match the path specified in the -binpath argument. | |
### Action dependent arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-newname &lt;name&gt;</td>
<td>A new name of a service.</td>
</tr>
<tr>
<td>-startuptype &lt;type&gt;</td>
<td>The startup type can be one of:</td>
</tr>
<tr>
<td></td>
<td>• auto</td>
</tr>
<tr>
<td></td>
<td>• manual</td>
</tr>
<tr>
<td></td>
<td>• disabled</td>
</tr>
<tr>
<td>-uid &lt;username&gt;</td>
<td>The username to use for a service.</td>
</tr>
<tr>
<td>-servicetype &lt;type&gt;</td>
<td>The type can be one of:</td>
</tr>
<tr>
<td></td>
<td>STRSCS – a StreamServer service</td>
</tr>
<tr>
<td></td>
<td>STRSDL – a StreamStudio service</td>
</tr>
<tr>
<td></td>
<td>STRSGW – a service gateway service.</td>
</tr>
<tr>
<td>-description &lt;description&gt;</td>
<td>A description of the service.</td>
</tr>
<tr>
<td>-binpath &lt;path&gt;</td>
<td>Path to the StreamServer executable, for example:</td>
</tr>
<tr>
<td></td>
<td>/var/streamserve/streamserve-5.4.0.GA.370/applications/streamserve/start</td>
</tr>
<tr>
<td>-argname &lt;name&gt;</td>
<td>The name of the startup argument.</td>
</tr>
<tr>
<td>-argvalue &lt;value&gt;</td>
<td>The value of the startup argument.</td>
</tr>
<tr>
<td>-applicationtype &lt;type&gt;</td>
<td>The type can be one of:</td>
</tr>
<tr>
<td></td>
<td>STRSCS – a StreamServer application</td>
</tr>
<tr>
<td></td>
<td>STRSDL – a StreamStudio application.</td>
</tr>
<tr>
<td></td>
<td>STRSGW – a service gateway application.</td>
</tr>
<tr>
<td></td>
<td>STRSCI – an Archiver application</td>
</tr>
<tr>
<td>-applicationversion &lt;version&gt;</td>
<td>The application version number.</td>
</tr>
<tr>
<td>-repeat &lt;number&gt;</td>
<td>Number of times to repeat the command. Default is 1.</td>
</tr>
<tr>
<td></td>
<td>Use only where it makes sense, for example with the action -getarg -argname &lt;argname&gt; argument.</td>
</tr>
</tbody>
</table>

### Optional arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-host &lt;hostname/ipnumber&gt;</td>
<td>If you do not specify this argument, localhost is used.</td>
</tr>
</tbody>
</table>
### Optional arguments and Description

<table>
<thead>
<tr>
<th>Optional arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-ipport &lt;port&gt;</code></td>
<td>IP port number. If you do not specify this argument, 28000 is used.</td>
</tr>
<tr>
<td><code>-timeout &lt;ms&gt;</code></td>
<td>Timeout in milliseconds. If you do not specify this argument, 5000 ms is used.</td>
</tr>
<tr>
<td><code>-cert &lt;filename&gt;</code></td>
<td>The client certificate file. The path to the file must be included in the filename.</td>
</tr>
<tr>
<td><code>-user &lt;username&gt;</code></td>
<td>The username for authentication. If you specify this argument, you must also specify the <code>-pass</code> argument.</td>
</tr>
<tr>
<td><code>-pass &lt;password&gt;</code></td>
<td>The password for authentication. If you specify this argument, you must also specify the <code>-user</code> argument.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Output version and exit.</td>
</tr>
<tr>
<td><code>-h</code></td>
<td>Display the readme file. (This user guide covers the information in the readme file).</td>
</tr>
</tbody>
</table>
Arguments for ss_territory utility

Required argument

- `action <action>` where `<action>` is one of the following:

<table>
<thead>
<tr>
<th>&lt;action&gt;</th>
<th>Description</th>
<th>Requires additional arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td>Gets the territory.xml for the specified application domain and creates a timestamp.</td>
<td><code>-territoryid &lt;territoryid&gt; or -territoryname &lt;territoryname&gt; -filename &lt;filename&gt;</code></td>
</tr>
<tr>
<td>update</td>
<td>Updates application domain information. By specifying the <code>-modtime</code> that was created when you used the get action, and the file has not been modified in the database since then, the update will succeed. By specifying <code>-force</code> instead of <code>-modtime</code>, the file will be overwritten without checking if it has been modified.</td>
<td><code>-territoryid &lt;territoryid&gt; or -territoryname &lt;territoryname&gt; -filename &lt;filename&gt; -force (Required only if the last modification time is out of sync) Optionally you can specify -modtime &lt;modtime&gt; If the -modtime argument is not specified, the -force argument must be specified.</code></td>
</tr>
<tr>
<td>delete</td>
<td>Deletes an application domain</td>
<td><code>-territoryid &lt;territoryid&gt; or -territoryname &lt;territoryname&gt;</code></td>
</tr>
<tr>
<td>create</td>
<td>Creates an application domain</td>
<td><code>-territoryid &lt;territoryid&gt; or -territoryname &lt;territoryname&gt; -filename &lt;filename&gt; -territoryversion &lt;version&gt; -territorydescription &lt;description&gt;</code></td>
</tr>
<tr>
<td>add_application</td>
<td>Adds an application to the application domain</td>
<td><code>-territoryid &lt;territoryid&gt; or -territoryname &lt;territoryname&gt; -appname &lt;name&gt; -apptype &lt;type&gt; -appvers &lt;vers&gt;</code></td>
</tr>
<tr>
<td>list_applications</td>
<td>Lists applications in an application domain.</td>
<td><code>-territoryid &lt;territoryid&gt; or -territoryname &lt;territoryname&gt;</code></td>
</tr>
<tr>
<td><strong>&lt;action&gt;</strong></td>
<td><strong>Description</strong></td>
<td><strong>Requires additional arguments</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>del_application</td>
<td>Deletes an application if it resides on the same machine as the management gateway. You can delete an application on a remote machine if you specify the application ID stored in the SER database.</td>
<td>-appname &lt;name&gt;</td>
</tr>
<tr>
<td>rename_application</td>
<td>Renames an application on the same machine as the management gateway. You must also run ss_scm -action newname &lt;name&gt; to synchronize the application name with the service name.</td>
<td>-appname &lt;name&gt;</td>
</tr>
<tr>
<td>display_node_info</td>
<td>Displays information on the current host</td>
<td></td>
</tr>
<tr>
<td>display_all_info</td>
<td>Displays information on all hosts that are registered in the StreamServe Enterprise Repository.</td>
<td></td>
</tr>
<tr>
<td>display_all_domain_names</td>
<td>Displays all application domains on the host. Optional: -territoryversion &lt;version&gt;</td>
<td></td>
</tr>
<tr>
<td>join</td>
<td>Moves an application between application domains.</td>
<td>-applicationid &lt;id&gt; -territoryid &lt;id&gt;</td>
</tr>
<tr>
<td>get_db_scripts</td>
<td>Generates the database scripts for the specified application domain. The scripts are zipped in a tgz file and stored in &lt;StreamServe installation&gt;/applications/managementgateway/etc on *NIX and &lt;StreamServe installation&gt;$Applications$Management\1.3\etc on Windows</td>
<td>-territoryname &lt;territoryname&gt; -db_type &lt;runtime/design&gt; If you specify runtime, scripts for the runtime database are created, if you specify design, scripts for the web content database are created.</td>
</tr>
</tbody>
</table>
### Arguments for ss_territory utility

**Command line utilities**

<table>
<thead>
<tr>
<th><code>&lt;action&gt;</code></th>
<th>Description</th>
<th>Requires additional arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_job_status</td>
<td>Displays the job status, e.g. if a database was created with <code>create_appdomain_db</code>. When everything in a log has been retrieved by a client, this action with the same job ID will fail.</td>
<td><code>-jobid &lt;jobid&gt;</code> &lt;br&gt;<code>-frompos &lt;pos&gt;</code> &lt;br&gt;<code>-maxlen &lt;max_bytes&gt;</code></td>
</tr>
<tr>
<td>rename_territory</td>
<td>Renames the application domain</td>
<td><code>-territoryname &lt;territoryname&gt;</code></td>
</tr>
<tr>
<td>app_in_sync</td>
<td>Checks if the <code>territory.xml</code> file for the application is in sync with the information in the StreamServe Enterprise Repository.</td>
<td><code>-appname &lt;name&gt;</code></td>
</tr>
<tr>
<td>deploy_domain_info</td>
<td>Exports the <code>territory.xml</code> to the specified application. You can only specify applications on the management gateway machine.</td>
<td><code>-appname &lt;name&gt;</code></td>
</tr>
<tr>
<td>create_appdomain_db</td>
<td>Creates a runtime database for the application domain. To check that the database was created, you must run the <code>get_job_status</code> action with the job ID that the <code>create_appdomain_db</code> action returned.</td>
<td><code>-dbuser &lt;user&gt;</code> &lt;br&gt;<code>-dbpass &lt;pass&gt;</code> &lt;br&gt;<code>-db_type &lt;runtime/design&gt;</code> &lt;br&gt;If you specify <code>runtime</code>, scripts for the runtime database are created, if you specify <code>design</code>, scripts for the web content database are created.</td>
</tr>
<tr>
<td>create_resource</td>
<td>Creates a resource in the StreamServe Enterprise Repository. The resource can be a StreamStudio portal, an Archiver node, Document Type etc.</td>
<td><code>-resource_version &lt;version&gt;</code> &lt;br&gt;<code>-filename &lt;path to resource file&gt;</code> &lt;br&gt;<code>-resource_name &lt;name&gt;</code> &lt;br&gt;<code>-content_type &lt;type&gt;</code> &lt;br&gt;<code>-strs_type &lt;type&gt;</code> &lt;br&gt;Optional arguments: &lt;br&gt;<code>-resource_description &lt;description&gt;</code> &lt;br&gt;<code>-territoryid &lt;ID&gt;</code> &lt;br&gt;<code>-nodeid &lt;ID&gt;</code> &lt;br&gt;<code>-content_encoding &lt;encoding&gt;</code></td>
</tr>
</tbody>
</table>
Arguments for `ss_territory` utility

### Command line utilities

**Update resource**

Updates a property on an existing resource. You can update the following properties:
- Territory ID
- Node ID
- Resource name
- Resource description
- Content encoding
- The resource data

*Action dependent arguments:*
- `-resource_id <ID>`

**Assign resource to territory relation**

Link a resource to an application domain. By using this action you can assign the same resource to several application domains.

*Action dependent arguments:*
- `-id <resource_id>`
- `-idto <domain_id>`

**Remove resource to territory relation**

Removes the link from the application domain to the resource.

*Action dependent arguments:*
- `-id <resource_id>`
- `-idto <domain_id>`

**Remove resource**

Removes a resource from the StreamServe Enterprise Repository.

*Action dependent arguments:*
- `-resource_id <ID>`

**Get resource data**

Retrieves the resource data.

*Action dependent arguments:*
- `-resource_id <ID>`
- `-filename <name>`

**Assign resource to application relation**

Links a resource to an application. For example, the Archiver application requires a link to the configuration resource.

*Action dependent arguments:*
- `-id <resource_id>`
- `-idto <domain_id>`

**Remove resource to application relation**

Removes the link from the application to the resource.

*Action dependent arguments:*
- `-id <resource_id>`
- `-idto <domain_id>`

### Action dependent arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-territoryid &lt;id&gt;</code></td>
<td>The application domain ID. Optionally use <code>-territoryname</code></td>
</tr>
</tbody>
</table>
## Arguments for ss_territory utility

### Command line utilities

- **-territoryname <name>**
  - The application domain name. Optionally use `-territoryid`.

- **-territoryversion <version>**
  - The application domain version.

- **-territorydescription <description>**
  - An application domain description.

- **-force**
  - Required for update action only if the last modification time is out of sync.

- **-modtime <modtime>**
  - The last modification time returned by the `get` action.

- **-applicationid <id>**
  - The application id.

- **-filename <filename>**
  - The file name to use for the application domain configuration file.

- **-appname <appname>**
  - The application name.

- **-appvers <version>**
  - The application version.

- **-apptype <type>**
  - The application type. The type can be one of:
    - STRSCS – a StreamServer application
    - STRSDL – a StreamStudio application.
    - STRSGW – a service gateway application.
    - STRSCI – an Archiver application.

- **-jobid <jobid>**
  - The job ID.

- **-dbuser <dbuser>**
  - Database administrator user.

- **-dbpass <dbpass>**
  - Database administrator user password.

- **-frompos <pos>**
  - Position in log file from which to retrieve content of the file.
  - To get all content specify 0.

- **-maxlen <maxbytes>**
  - Max number of bytes to retrieve from the log file. To get all content specify a high number, e.g. 60000.

- **-resource_version <version>**
  - A string of your choice that represents the version of the resource, for example 1.0 or Alpha 0.1.

- **-filename <path to resource file>**
  - The path to the file containing the resource data.

- **-resource_name <name>**
  - The name of the resource.

- **-content_type <type>**
  - A general content type, for example application/x-streamserve.com-webportal.
### Arguments for ss_territory utility

#### Command line utilities

For more optional arguments, see *Arguments for ss_scm utility* on page 8.

<table>
<thead>
<tr>
<th>Action dependent arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-strs_type &lt;type&gt;</td>
<td>A specific content type, for example application/x-streamserve.com-streamstudio</td>
</tr>
<tr>
<td>-resource_id &lt;id&gt;</td>
<td>The resource ID given when creating the resource.</td>
</tr>
<tr>
<td>-id &lt;id&gt;</td>
<td>The resource ID when assigning and removing resources to and from domains and applications.</td>
</tr>
<tr>
<td>-idto &lt;domain_id&gt;</td>
<td>The application domain when assigning and removing resources to and from it.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-resource_description &lt;description&gt;</td>
<td>A description of the resource</td>
</tr>
<tr>
<td>-territoryid &lt;ID&gt;</td>
<td>Not used in this release.</td>
</tr>
<tr>
<td>-nodeid &lt;ID&gt;</td>
<td>The physical server hosting the resource.</td>
</tr>
<tr>
<td>-content_encoding &lt;encoding&gt;</td>
<td>Specifies if the data is compressed, for example application/x-gzip. Leave empty for plain data.</td>
</tr>
</tbody>
</table>
Arguments for ss_deploy utility

Required arguments

-`action` `<action>` where `<action>` is one of the following:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
<th>Requires additional arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploy</td>
<td>Deploys a package with a specified platform to an application.</td>
<td><code>-physicalplatform</code> <code>&lt;platform&gt;</code> <code>-package</code> <code>&lt;package&gt;</code></td>
</tr>
<tr>
<td>deploy_application_config_files</td>
<td>Use for Archiver and Service Gateway applications. This action requires that a working directory has been created for the application. The action copies all template files the application depends on to the working directory. For example, if you want application specific log settings, a specific logmanager.xml is required in the application’s working directory. <strong>Note:</strong> This action only takes the <code>-appname</code> argument.</td>
<td><code>-appname</code> <code>&lt;name&gt;</code></td>
</tr>
</tbody>
</table>

Required arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-appname</code> <code>&lt;name&gt;</code></td>
<td>The StreamServer application to deploy to.</td>
</tr>
<tr>
<td><code>-physicalplatform</code> <code>&lt;physicalplatform&gt;</code></td>
<td>The physical platform to deploy. <strong>Note:</strong> A physical platform must exist in the export package.</td>
</tr>
<tr>
<td><code>-package</code> <code>&lt;package&gt;</code></td>
<td>The package file to deploy. The package file is the one exported from Design Center.</td>
</tr>
</tbody>
</table>
Optional arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-projectname &lt;name&gt;</td>
<td>The name of the Project, normally the export file without the extension. If this argument is not specified the name of the exportfile is used.</td>
</tr>
<tr>
<td>-projectlabel &lt;label&gt;</td>
<td>The label to use, specify revision from CVS or other label.</td>
</tr>
</tbody>
</table>

For more optional arguments, see *Arguments for ss_scm utility* on page 8.
Examples

The following examples are for Windows where the commands are run from $StreamServe_installation\Platform\Core\1.2\bin

For UNIX, instead of running the .exe files, you browse to $STRS_LOCATION/streamserve/applications/managementgateway/bin and run for example ./start ss_territory <args>

Creating an application domain

Prerequisites
You must have access to a territory.xml whose settings reflect your environment. You can for example use an existing territory.xml file and modify it according to your needs or create a new one from scratch.

Command
ss_territory.exe -user USERID -pass PASSWD -cert C:\Program Files\StreamServe\Platform\Core\1.2\bin\security\certificatestore\trusted\authorities\streamserve.ca.crt -action create -territoryname MYAPPDOMAIN -filename "territory.xml" -territoryversion 1.0 -territorydescription "My New Application Domain"

Modifying an existing application domain

Prerequisites
You must have access to a territory.xml whose settings have been modified to reflect your environment. You can for example use an existing territory.xml file and modify it according to your needs or create a new one from scratch.

Command
ss_territory.exe -user USERID -pass PASSWD -cert C:\Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore\trusted\authorities\streamserve.ca.crt -action update -territoryname MYAPPDOMAIN -filename territory.xml -territoryversion 1.0 -territorydescription "My Updated Application Domain" -force

Creating a StreamServer application

Prerequisites
An application domain must be created.
Creating a service for a StreamServer application

Prerequisites
A StreamServer application must be created.

Command
```
ss_territory.exe -user USERID -pass PASSWD -cert \Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore \trusted\authorities\streamserve.ca.crt -action add_application -territoryname MYAPPDOMAIN -appname MYAPPLICATION -appvers 1.0 -apptype STRSCS
```

Deploying an export package to an application

Prerequisites
- A service for a StreamServer application must be created.
- A working directory containing an export package must exist.

Command
```
ss_deploy.exe -user USERID -pass PASSWD -cert \Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore \trusted\authorities\streamserve.ca.crt -action deploy -appname MYSTRSSERVICE -physicalplatform MYPHYSICALPLATFORM -package "MYCONFIGURATION.export" -projectlabel 1.0
```

Modifying a StreamServer service

Prerequisites
A StreamServer service with a deployed export package must exist.

Command
```
ss_scm.exe -user USERID -pass PASSWD -cert \Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore \trusted\authorities\streamserve.ca.crt -servicename MYSTRSSERVICE -action newarg -argname "-wd" -argvalue "C:\Management Gateway\1.0\root\Applications\myapp"
```
Starting a StreamServer service

**Prerequisites**
A StreamServer service with a deployed export package must exist.

**Command**
```
ss_scm.exe -user USERID -pass PASSWD -cert "C:\Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore\trusted\authorities\streamserve.ca.crt" -servicename MYSTRSSERVICE -action start
```

Checking if a StreamServer service is running

**Prerequisites**
A StreamServer service with a deployed export package must exist.

**Command**
```
ss_scm.exe -user USERID -pass PASSWD -cert "C:\Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore\trusted\authorities\streamserve.ca.crt" -servicename MYSTRSSERVICE -action isrunning
```

Stopping a StreamServer service

**Prerequisites**
A StreamServer service with a deployed export package must exist.

**Command**
```
ss_scm.exe -user USERID -pass PASSWD -cert "C:\Program Files\StreamServe\Platform\Core\1.3\bin\security\certificatestore\trusted\authorities\streamserve.ca.crt" -servicename MYSTRSSERVICE -action stop
```

Creating a web portal and assign it to a domain

1. Create an XML file, for example `MyPortal.xml`:
```
<?xml version="1.0 encoding=“utf-8” ?>
<ResourceConfig>
    <!--PortalRoot is used by Management Gateway and Control Center when updating domain information for portal-->
    <PortalRoot>C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps</PortalRoot>
</ResourceConfig>
```

or for unmanaged (manual deploy to Java application server)
```
<?xml version="1.0 encoding=“utf-8” ?>
```
<ResourceConfig>
<!--Used by Control Center for display. Not mandatory-->
<ManualDeployPath>C:\Management Gateway\etc\unmanagedportals</ManualDeployPath>
</ResourceConfig>

2. Create a portal resource with the following command:

```
```

3. Create a relation between the portal resource and an application domain with the following command:

```
ss_territory.exe -action assign_resource_to_territory_relation -id 80652796-7020-4740-9E91-8F3A6764CC8D -idto <appdomain_id>
```

### Creating an Archiver node

To create an Archiver node in the application domain, you perform the steps as in *Creating a web portal and assign it to a domain* on page 22 but the XML file contains a security profile holding a standard connection profile found in `territory.xml`.

**Note:** The `-content_type` argument is in this case `application/x-streamserve.com-archive` and the `-strs_type` argument is `application/x-streamserve.com-securityprofiles`.

### Creating an Archiver application

The XML file you need for creating an Archiver application (as the `MyPortal.xml` in *Creating a web portal and assign it to a domain* on page 22) can be obtained by creating and configuring an Archiver application in Control Center and then retrieve the file from the working directory.

Then create an application with the `ss_territory` and `ss_scm` utilities (the `-apptype` argument is set to `STRSCI`) and create a resource and a relation as in *Creating a web portal and assign it to a domain* on page 22.

**Note:** The `-content_type` argument is in this case `application/x-streamserve.com-configuration` and the `-strs_type` argument is `application/x-streamserve.com-generic-app-configuration`
Troubleshooting

The utilities does not start

Solution
Check that there is an environment variable called STRS_LOCATION (in uppercase letters) that points to the Streamserver installation directory. Remember that in many UNIX dialects you must export the variable after you have defined it.

Prompt displayed for realm user name

(or e.g. Error: System specific error code:146)
This is probably because the management gateway connection is not working.

Solution
Check that:
• You have specified correct user name and password for the management gateway
• The management gateway is started.
• The -ipport argument specifies the correct port for your management gateway.

HTTP listener error displayed

Error: HTTPListener(scm): The file or a script interpreter is not a regular file, execute permission is denied for the file or ascript or ELF interpreter, the file system is mounted noexec or search permission is denied on a component of the path prefix of filename or the name of a script interpreter.

Solution
Make sure you specify the correct -binpath value when creating the service, i.e. the path to the start binary. For example -binpath /var/streamserve/streamserve-5.3.0.GA.200/applicaitons/streamserver/start

Note: The file must be included in the path, not just the path to the directory where the file is stored.