Namespace Service User's Guide

Version 1.0.3

Last updated: August 26, 2011

Table of Contents

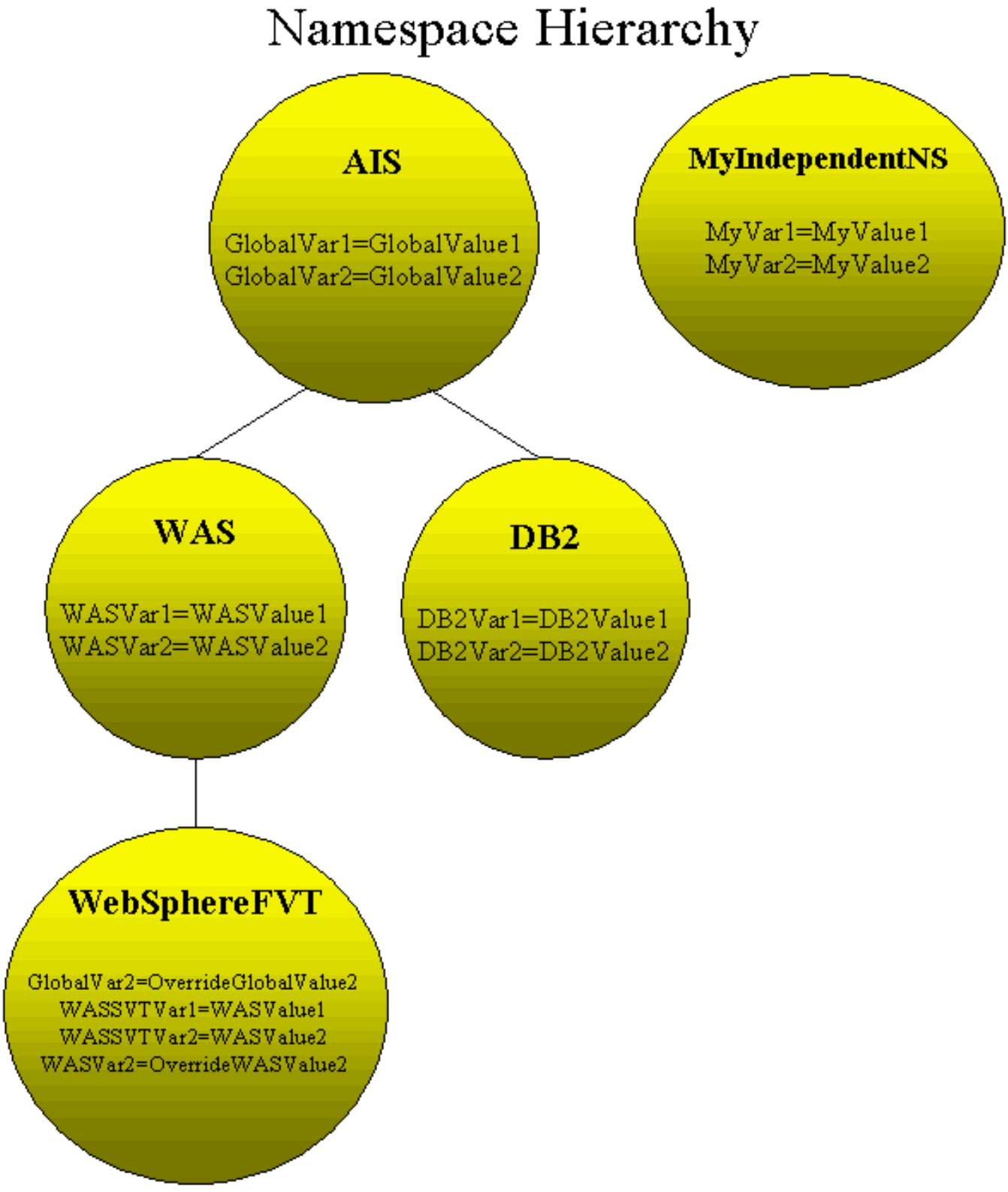
Overview

The Namespace service is an external STAF service written in Java. The purpose of the Namespace service is to provide a namespace hierarchy for storing and retrieving a persistent repository of variables. This service allows the creation of namespaces and allows you to set key/value pairs (e.g. variables) in a namespace. Namespaces may inherit variables from a parent namespace, thus creating a namespace hierarchy. Variable look-ups will be done within a namespace scope. If a

variable cannot be found in the given namespace, resolution will be attempted in the parent namespace, and so on up the hierarchy.

Unlike the VAR service, any variables set in a namespace will persist across stops and restarts of STAF with no additional steps required by the user. This will be done by immediately updating an XML file when any updates are made to namespaces.

Here's an example of a namespace hierarchy:



Installation and Configuration

1. Install Java 1.3 or later.

2. Install STAF 3.0.0 or later by following the installation instructions in the STAF documentation.

3. Install the Namedpace service:

   Download the NamespaceV103.zip/tar file from Get STAF Services into a local directory (e.g. C:/STAF/services or /usr/local/staf/services) and extract it.

4. Configure the Namespace service:

   Add the following statement to your staf.cfg file:

   ```
   SERVICE <Service Name> LIBRARY JSTAF EXECUTE <Service Jar File Name> \
                          PARMS "[DIRECTORY <Directory>] [FILENAME <FileName>]"
   ```

   where:

   - `<Service Name>` is the name by which the Namespace service will be known on this machine.
   - `<Service Jar File Name>` is the fully-qualified name of the Namespace.jar file.
   - `DIRECTORY` specifies the fully-qualified path where the file containing the namespaces data will be stored. This parameter is optional and defaults to `{STAF/DataDir}/service/`.
   - `FILENAME` specifies the name of the file containing the namespaces data. This parameter is optional and defaults to Namespaces.xml.

   Examples:

   ```
   SERVICE Namespace LIBRARY JSTAF EXECUTE {STAF/Config/STAFRoot}/services/
   namespace/Namespace.jar

   SERVICE NS LIBRARY JSTAF EXECUTE C:/STAF/services/namespace/Namespace.jar \
           PARMS "DIRECTORY C:/test/namespace FILENAME NS.xml"
   ```

   Or, you can dynamically add the Namespace service using the SERVICE service's `ADD SERVICE` request.

In most cases, the way that you would use this service is that you would designate a system as the Namespace Server. That system would have STAF installed, plus the Namespace service with the `SERVICE` configuration statement in the configuration file that is shown above. Assuming that the machine name for the Namespace server is server1 and the registered service name is Namespace, an example of the command line interface to create a namespace is:

```
STAF server1 Namespace CREATE NAMESPACE AIS DESCRIPTION "Global AIS Namespace"
```

Request Syntax

The Namespace service provides the following requests:

- CREATE - creates a new namespace in which variables may be created
- MODIFY - modifies the description or parent for an existing namespace
- DELETE - deletes a namespace or deletes a variable from a namespace
- LIST - returns information about all registered namespaces or returns a list of variables defined within a namespace or returns a list of the settings for the service
- QUERY - returns information about a single namespace or provides a tree hierarchy view for a namespace, including all it's child namespaces
- SET - creates or sets variable(s) within a namespace
- GET - gets the value for a variable within a namespace
- VERSION - displays the version of the Namespace service
- HELP - provides the help syntax for the Namespace service

CREATE

The `CREATE` command creates a new namespace in which variables may be created. The namespace may optionally inherit variables from another, already existing, namespace. If the namespace has no parent it will inherit no variables.

The only reserved name for a namespace is the string NONE, case insensitive. This string will be used to represent a root namespace -- a namespace that does not have a parent.

Syntax

```
CREATE NAMESPACE <Name> DESCRIPTION <Description> [PARENT <Name>]
```

`NAMESPACE` specifies the name of the namespace to create. This can be any string (except for `NONE`, case insensitive, which is reserved). This option will resolve variables.

`DESCRIPTION` specifies a textual description of the new namespace.

`PARENT` specifies the name of the parent namespace that this namespace should inherit variables from. The parent namespace must exist. If the `PARENT` option is not specified, the new namespace will have no parent and will not inherit any variables. This can also be achieved by specifying `NONE` (case-insensitive). This option is optional. This option will resolve variables.

Security

This command requires trust level 3.

Results

Upon successful return, the result buffer will be empty.

Examples:

- **Goal:** Create a new namespace named "AIS" which exists as the top of a namespace hierarchy.:

  ```
  CREATE NAMESPACE AIS DESCRIPTION "Global AIS Namespace"
  ```

- **Goal:** Create a new namespace named "WAS" which inherits variables from the "AIS" namespace:

  ```
  CREATE NAMESPACE WAS PARENT AIS DESCRIPTION "Global WebSphere Namespace"
  ```

MODIFY

The `MODIFY` command may be used to modify the parent and/or description of an existing namespace.

Syntax

```
MODIFY NAMESPACE <Name> [DESCRIPTION <Description>] [PARENT <Name>]
```

`NAMESPACE` specifies the name (case-insensitive) of the namespace to be modified. This option will resolve variables.

`DESCRIPTION` specifies the new description for this namespace.

`PARENT` specifies the name (case-insensitive) of the new parent for this namespace. This option will resolve variables. Be aware that any children of this namespace will keep this namespace as their parent and will move to the new hierarchy location as well. To specify that the namespace should be modified to have no parent (a root namespace), then specify `NONE` (case-insensitive).

At least one of the `DESCRIPTION` or `PARENT` options must be specified, but both may specified as well.

Security

This command requires trust level 3.

Results

Upon successful return, the result buffer will be empty.

Examples

- **Goal:** Change the description for namespace "AIS" to be "My root namespace":

  ```
  MODIFY NAMESPACE AIS DESCRIPTION "My root namespace"
  ```

- **Goal:** Change the parent for namespace "WebSphereSVT" to be "WAS":

  ```
  MODIFY NAMESPACE WebSphereSVT PARENT WAS
  ```

- **Goal:** Change the description and the parent for namespace "WAS", specifying None to indicate that the WAS namespace will no longer have a parent:

  ```
  MODIFY NAMESPACE WAS DESCRIPTION "WAS Namespace" PARENT None
  ```

DELETE

The DELETE command may be used to delete one or more variables from a namespace or to delete a namespace.

Syntax

```
DELETE NAMESPACE <Name> <VAR <Key>... | CONFIRM>
```

NAMESPACE specifies the name (case-insensitive) of a namespace. This option will resolve variables.

VAR specifies the key (case-insensitiveI) of a variable to delete from the namespace. This option may appear more than one time. This option will resolve variables.

CONFIRM indicates that you want to delete the namespace and any variables directly contained within it. If the namespace has any child namespaces, their parents will be changed to be the parent of the deleted namespace.

You cannot specify both the VAR and CONFIRM options.

Security

This command requires trust level 4 if deleting a namespace or trust level 3 if deleting a variable from a namespace.

Results

Upon successful return, the result buffer will be empty.

Examples

- **Goal:** Delete the variable with key "GlobalVar1" in the "AIS" namespace:

  ```
  DELETE NAMESPACE AIS VAR GlobalVar1
  ```

- **Goal:** Delete multiple variables (with keys "WASVar1" and "WASVar2") in the "WAS" namespace:

  ```
  DELETE NAMESPACE WAS VAR WASVar1 VAR WASVar2
  ```

- **Goal:** Delete a leaf namespace "WebSphereSVT":

  ```
  DELETE NAMESPACE WebSphereSVT CONFIRM
  ```

- **Goal:** Delete an intermediary namespace "WAS"

```
DELETE NAMESPACE WAS CONFIRM
```

LIST

The LIST command may be used to list all namespaces or variables defined within a namespace, or the settings for the service, depending on the option(s) specified.

Syntax

```
LIST [NAMESPACES | <NAMESPACE <Name> [ONLY]> | SETTINGS]
```

`NAMESPACES` specifies to list all namespaces. This is the default.

`NAMESPACE` specifies the name (case-insensitive) of a namespace for which its variables will be listed. This option will resolve variables.

`ONLY` specifies to list only the variables defined directly in the specified namespace. None of the variables from any parents of this namespace will be listed. If the `ONLY` option is not specified, all variables from the given namespace will be listed as well as any variables from parent namespaces which were not overridden n this namespace.

`SETTINGS` specifies to list the settings for the service.

If neither the `NAMESPACE` nor `NAMESPACES` nor `SETTINGS` option is specified, a list of the namespaces will be returned.

Security

This command requires trust level 2.

Results

Upon successful return,

- The result buffer for a `"LIST"` or `"LIST NAMESPACES"` request will contain a marshalled `<List>` of `<Map: STAF/Service/Namespace/NamespaceInfo>` which represents a list of all namespaces. The map is defined as follows:

Definition of map class STAF/Service/Namespace/NamespaceInfo			
Description: This map class represents a namespace.			
Key Name	**Display Name**	**Type**	**Format / Value**
name	Name	`<String>`	
description	Description	`<String>`	
parent	Parent	`<String> | <None>`	

Notes:
1. "Name" is the name of a namespace.
2. "Description" is a description of a namespace.
3. "Parent" is the name of the parent namespace. If the namespace has no parent, the parent will be `<None>`.

- The result buffer for a `"LIST NAMESPACE"` request will contain a marshalled `<List>` of <Map:STAF/Service/Namespace/VarInfo> which represents a list of variables in the specified namespace. The map is defined as follows:

Definition of map class STAF/Service/Namespace/VarInfo			
Description: This map class represents a variable defined in a namespace.			
Key Name	**Display Name**	**Type**	**Format / Value**
key	Key	`<String>`	
value	Value	`<String>`	
namespace	Namespace	`<String>`	

Notes:
1. "Key" is the key of the variable.
2. "Value" is the value of the variable.
3. "Namespace" is the name of the namespace that the variable is defined in.

- The result buffer for a `"LIST SETTINGS"` request will contain a marshalled `<List>` of <Map:STAF/Service/Namespace/Settings> which represents a list of the settings for the service. The map is defined as follows:

Definition of map class STAF/Service/Namespace/Settings			
Description: This map class represents the settings for the service.			
Key Name	**Display Name**	**Type**	**Format / Value**
directory	Directory	`<String>`	
filename	File Name	`<String>`	

Notes:
1. "Directory" is the directory where the file containing the data for the Namespace service is located.
2. "File Name" is the name of the file containing the data for the Namespace service.

Examples

For the following examples, assume that the namespaces are defined as shown in the example XML file in the Namespaces XML File section.

- **Goal:** List all of the namespaces.

  ```
  LIST NAMESPACES
  ```

  Result: If the request is submitted from the command line, the result in the table format could look like:

  ```
  Name            Description                Parent
  ```

```
-------------- ------------------------- ------
AIS            Global AIS Namespace      <None>
DB2            Global DB2 Namespace      AIS
MyIndependentNS User Independent Namespace <None>
WAS            Global WebSphere Namespace AIS
WebSphereSVT   WebSphere SVT Namespace
```

- **Goal:** List only the variables defined in the WebSphereSVT namespace:

  ```
  LIST NAMESPACE WebSphereSVT ONLY
  ```

  Result: If the request is submitted from the command line, the result in the table format could look like:

  ```
  Key        Value                Namespace
  ---------- -------------------- ------------
  GlobalVar2 OverrideGlobalValue2 WebSphereSVT
  WASSVTVar1 WASSVTValue1         WebSphereSVT
  WASSVTVar2 WASSVTValue2         WebSphereSVT
  WASVar2    OverrideWASValue2    WebSphereSVT
  ```

- **Goal:** List the variables defined in the WebSphereSVT namespace, as well as any variables from parent namespaces that were not overridden in this namespace :

  ```
  LIST NAMESPACE WebSphereSVT
  ```

  Result: If the request is submitted from the command line, the result in the table format could look like:

  ```
  Key        Value                Namespace
  ---------- -------------------- ------------
  GlobalVar2 OverrideGlobalValue2 WebSphereSVT
  WASSVTVar1 WASSVTValue1         WebSphereSVT
  WASSVTVar2 WASSVTValue2         WebSphereSVT
  WASVar2    OverrideWASValue2    WebSphereSVT
  WASVar1    WASValue1            WAS
  GlobalVar1 GlobalValue1         AIS
  ```

- **Goal:** List the settings for the Namespace service:

  ```
  LIST SETTINGS
  ```

  Result: If the request is submitted from the command line, the result could look like:

  ```
  Directory: C:\STAF\data\STAF\service\namespace
  File Name: Namespaces.xml
  ```

QUERY

The QUERY command may be used to query information about a single namespace or to view the namespace hierarchy

from the given namespace down the hierarchy through all leaf namespaces.

Syntax

```
QUERY NAMESPACE <Namespace> [TREE]
```

NAMESPACE specifies the name (case-insensitive) of the namespace to query. This option will resolve variables.

TREE indicates to return the tree hierarchy view from the given namespace down through all child leaf namespaces.

Security

This command requires trust level 2.

Results

Upon successful return:

- The result buffer for a "QUERY NAMESPACE" request (without the TREE option) will contain a marshalled <Map:STAF/Service/Namespace/Query> which represents detailed information about the namespace. The map is defined as follows:

Definition of map class STAF/Service/Namespace/Query			
Description: This map class represents detailed information about a namespace.			
Key Name	**Display Name**	**Type**	**Format / Value**
name	Name	<String>	
description	Description	<String>	
parent	Parent	<String> | <None>	
children	Children	<List> of <String>	
Notes: 1. "Children" will contain a list of the names of the immediate child namespaces in this namespace, if any.			

- The result buffer for a "QUERY NAMESPACE TREE" request will contain a marshalled <Map:STAF/Service/Namespace/QueryTree> which represents the namespace and its children in a hierarchical manner. The map is defined as follows:

Definition of map class STAF/Service/Namespace/QueryTree			
Description: This map class represents the namespace and its children in a hierarchical manner.			
Key Name	**Display Name**	**Type**	**Format / Value**
name	Name	<String>	
children	Children	<List> of <Map:STAF/Service/Namespace/QueryTree>	

Examples

For the following examples, assume that the namespaces are defined as shown in the example XML file in the Namespaces XML File section.

- **Goal:** Show detailed information about namespace AIS.

  ```
  QUERY NAMESPACE AIS
  ```

  Result: If the request is submitted from the command line, the result, in the default format, could look like:

  ```
  {
    Name       : AIS
    Description: Global AIS Namespace
    Parent     :
    Children   : [
      DB2
      WAS
    ]
  }
  ```

- **Goal:** Show hierarchical information about the namespaces contained in namespace AIS.

  ```
  QUERY NAMESPACE AIS TREE
  ```

  Result: If the request is submitted from the command line, the result, in the default format, would look like:

  ```
  {
    Name    : AIS
    Children: [
      {
        Name    : DB2
        Children: []
      }
      {
        Name    : WAS
        Children: [
          {
            Name    : WebSphereSVT
            Children: []
          }
        ]
      }
    ]
  }
  ```

SET

The SET command creates or modifies one or more variables within a namespace.

Syntax

```
SET VAR <Key=Value> [VAR <Key=Value>] NAMESPACE <Name>
```

VAR specifies the key and value of the variable to set, separated by "=". This parameter may appear 1 or more times.

NAMESPACE specifies the name (case-insensitiveI) of the namespace in which to create the variable(s). This option will resolve variables.

Security

This command requires trust level 3.

Results

On a successful return, the result buffer will be empty.

Examples

- **Goal:** Set a new variable in the "AIS" namespace.

  ```
  SET VAR GlobalVar2=GlobalValue2 NAMESPACE AIS
  ```

- **Goal:** Change the value of the variable withe key "GlobalVar2" in the "AIS" namespace.

  ```
  SET VAR GlobalVar2=MyNewValue NAMESPACE AIS
  ```

- **Goal:** Set multiple variables in the "WebSphereSVT" namespace.

  ```
  SET VAR WASSVTVar1=WASSVTValue1 VAR GlobalVar2=OverrideGlobalValue2 NAMESPACE
  WebSphereSVT
  ```

GET

The GET command gets the value of a particular variable within a namespace. If the variable does not exist in the specified namespace, the parent namespace will be checked and so on up the hierarchy. The value for the first instance of the variable found will be returned.

Syntax

```
GET VAR <Key> NAMESPACE <Name>
```

VAR specifies the key (case-insensitive) of the variable that you want to get the value of. This option will resolve variables.

NAMESPACE specifies the name (case-insensitive) of the namespace in which to begin looking for the variable. This option will resolve variables.

Security

This command requires trust level 2.

Results

On a successful return, the result buffer will contain the value for the first instance of the variable found.

Examples

For the following examples, assume that the namespaces are defined as shown in the example XML file in the Namespaces XML File section.

- **Goal:** Get the value for the variable with key "GlobalVar2" starting in the "AIS" namespace.

  ```
  GET VAR GlobalVar2 NAMESPACE AIS
  ```

  Result:

  ```
  GlobalValue2
  ```

- **Goal:** Get the value for the variable with key "GlobalVar2" starting in the "WebSphereSVT" namespace.

  ```
  GET VAR GlobalVar2 NAMESPACE WebSphereSVT
  ```

  Result:

  ```
  OverrideGlobalValue2
  ```

- **Goal:** Get the value for the variable with key "GlobalVar1" starting in the "WebSphereSVT" namespace.

  ```
  GET VAR GlobalVar1 NAMESPACE WebSphereSVT
  ```

  Result:

  ```
  GlobalValue1
  ```

VERSION

VERSION displays the Namespace Service version.

Syntax

```
VERSION
```

Security

This request requires at least trust level 1.

Results

The result is the version number of the Namespace service.

Examples

- **Goal:** Display the version of the Namespace service on machine server1:

  ```
  STAF server1 Namespace VERSION
  ```

  Output:

  ```
  1.0.3
  ```

HELP

HELP displays the request options and how to use them.

Syntax

```
HELP
```

Security

This request requires at least trust level 1.

Results

The result buffer contains the Help messages for the request options for the Namespace service.

Examples

- **Goal:** Display the syntax for the Namespace service on machine server1:

  ```
  STAF server1 Namespace HELP
  ```

  Output:

  ```
  Namespace Service Help

  CREATE  NAMESPACE <Name> DESCRIPTION <Description> [PARENT <Name>]
  MODIFY  NAMESPACE <Name> [DESCRIPTION <Description>] [PARENT <Name>]
  DELETE  NAMESPACE <Name> < VAR <Key>... | CONFIRM >
  LIST    [NAMESPACES | <NAMESPACE <Name> [ONLY]> | SETTINGS]
  QUERY   NAMESPACE <Name> [TREE]
  SET     VAR <Key=Value> [VAR <Key=Value>]... NAMESPACE <Name>
  ```

```
GET     VAR <Key> NAMESPACE <Name>
VERSION
HELP
```

Error Code Reference

In addition to the common STAF return codes, the following service return codes are defined for the Namespace service:

Error Code	Meaning	Comment
4001	Data storage error	An error occurred saving namespaces data to persistent storage. Additional information about the error is put into the STAF Result.

Service Logging

The Namespace service maintains a global log where it writes an entry when:

- The Namespace service is registered, logs a message with log level "Start".
- A CREATE, MODIFY, DELETE, or SET request is performed successfully, logs a message with log level "Info". The message includes information about the originator of the request and the actual request submitted in the following format:

  ```
  [<Orig machine> <Orig handle name> <Orig handle] <Request>
  ```

- A Namespace service error occurs (such as when an error occurs reading or writing the Namespaces XML file), logs a message with log level "Error" and detailed information about the error in the message.
- The Namespace service is unregistered, logs a message with log level "Stop"

The logname for the Namespace service is the name under which the service is registered.

Here is an example of what a Namespace service log (with the Namespace service's registered name begin NS) could look like shown via a request from the command line in the table format:

```
C:\>STAF local LOG QUERY GLOBAL LOGNAME NS
Response
--------
20051016-16:43:24 Start Initialized the NS service
20051016-16:44:47 Info  [local://local STAX/Job/1 27] CREATE NAMESPACE WebSpher
                        eSVT DESCRIPTION WebSphere SVT Namespace PARENT WAS
20051016-16:44:47 Info  [local://local STAX/Job/1 27] SET NAMESPACE WebSphereSV
```

```
                        T VAR WASSVTVar1=WASSVTValue1 VAR WASSVTVar2=WASSVTValu
                        e2 VAR WASVar2=OverrideWASValue2 VAR GlobalVar2=Overrid
                        eGlobalValue2
20051016-16:44:47 Info  [local://local STAX/Job/1 27] CREATE NAMESPACE DB2 DESC
                        RIPTION Global DB2 Namespace PARENT AIS
20051016-16:44:48 Info  [local://local STAX/Job/1 27] SET NAMESPACE DB2 VAR DB2
                        Var1=DB2Value1 VAR DB2Var2=DB2Value2
20051016-16:44:52 Info  [local://local STAX/Job/1 27] DELETE NAMESPACE WEBSpher
                        eSVT CONFIRM
20051016-16:44:53 Info  [local://local STAX/Job/1 27] DELETE NAMESPACE DB2 CONF
                        IRM
20051016-16:46:42 Stop  Terminating the NS service
```

Service Data Storage

Unlike the VAR service, any variables set in a namespace will persist across stops and restarts of STAF with no additional steps required by the user. This will be done by immediately updating an XML file when any updates are made to namespaces.

Namespaces DTD

The Namespaces Document Type Definition (DTD) is defined as follows and will be used to validate the Namespaces XML file. Parent-child relationships are represented by nested <namespace> elements.

```
<!--
   STAF Namespaces Document Type Definition (DTD)

   This DTD module is identified by the SYSTEM identifier:

     SYSTEM 'resources/Namespaces.dtd'

-->

<!ELEMENT namespaces     (namespace)*>

<!ELEMENT namespace      (var*, namespace*)>
<!ATTLIST namespace
          name           CDATA    #REQUIRED
          description    CDATA    #REQUIRED
>

<!ELEMENT var            (#PCDATA)>
<!ATTLIST var
          key            CDATA    #REQUIRED
          value          CDATA    #REQUIRED
>
```

Namespaces XML File

The Namespace service stores namespaces and their variables in an XML file. By default the file is named `Namespaces.xml` and is located in directory `{STAF/DataDir}/service/<Service Name (lower-case)>`. To override the storage location, you can specify the DIRECTORY and/or FILENAME options in the PARMS value when registering the Namespace service. See the Installation and Configuration section for more information on setting these parameters.

The Namespace service immediately updates this XML file at the end of each successful CREATE, MODIFY, DELETE, or SET request.

For example, on Windows, the Namespaces xml file will be stored in C:\STAF\data\STAF\service\ns\Namespacees.xml if the Namespace service is registered as NS with no parameters and STAF is installed at C:\STAF using the default STAF instance name of STAF.

Here's an example of a Namespaces xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE namespaces SYSTEM "resources/Namespaces.dtd">
<namespaces>
  <namespace description="Global AIS Namespace" name="AIS">
    <var key="GlobalVar1" value="GlobalValue1"/>
    <var key="GlobalVar2" value="GlobalValue2"/>
    <var key="GlobalVar3" value="GlobalValue3"/>
    <namespace description="Global DB2 Namespace" name="DB2">
      <var key="DB2Var1" value="DB2Value1"/>
      <var key="DB2Var2" value="DB2Value2"/>
    </namespace>
    <namespace description="Global WebSphere Namespace" name="WAS">
      <var key="WASVar1" value="WASValue1"/>
      <var key="WASVar2" value="WASValue2"/>
      <namespace description="WebSphere SVT Namespace" name="WebSphereSVT">
        <var key="GlobalVar2" value="OverrideGlobalValue2"/>
        <var key="WASSVTVar1" value="WASSVTValue1"/>
        <var key="WASSVTVar2" value="WASSVTValue2"/>
        <var key="WASVar2" value="OverrideWASValue2"/>
      </namespace>
    </namespace>
  </namespace>
  <namespace description="User Independent Namespace" name="MyIndependentNS">
    <var key="MyVar1" value="MyValue1"/>
    <var key="MyVar2" value="MyValue2"/>
  </namespace>
</namespaces>
```

Backup XML Files

For backup purposes, the Namespace service makes a temporary copy of the Namespaces xml file at the following times:

- When the Namespace service is registered (`Phase = init`)
- Immediately before a namespace is deleted (`Phase = delete`)
- When the Namespace service is unregisterd (`Phase = term`)

These files are temporary because they are stored in STAF's temporary directory (`{STAF/DataDir}/tmp`). STAF's temporary directory and all of its contents are deleted whenever STAF is restarted.

The directory where these temporary files are stored is:

```
{STAF/DataDir}/tmp/service/<Service Name (lower-case)>.
```

For example, on Windows, this directory could be `C:\STAF\data\STAF\tmp\service\ns`.

The format of the names of the copied Namespace xml files will be:

```
<Namespace File Name>.<Phase>.<Current Time (milli-seconds)>
```

So, for example, the temporary directory for the Namespace service could contain files such as:

```
Namespaces.xml.init.1129499304537
Namespaces.xml.delete.1129499341040
Namespaces.xml.delete.1129499341200
Namespaces.xml.term.1129499418842
```

Also, a backup of the Namespaces xml file named `<Namespace FileName>.backup` is created immediately before the Namespace service updates the Namespaces.xml file during a CREATE, MODIFY, DELETE, or SET request. This backup file is located in the namespace directory along with the Namespaces xml file.