Nirva NDAS Service

Document Version: 1.00

Table of Contents

Overview

The Nirva Directory Access Service (NDAS) is a service for Nirva that allows Nirva applications to authenticate their users against an LDAP server.

Once NDAS has authenticated a user, it can then assign Nirva permissions to the user session.

Compatibility

NDAS has been written to be generic, i.e. it should function with any directory server that conforms to the LDAP specifications. It has been successfully tested with the following LDAP servers:

- Active Directory (from Microsoft)
- OpenLDAP servers (open source)

Configuration

Overview

In many cases, a single LDAP server will be sufficient for all applications. However, NDAS allows multiple LDAP servers to be configured so that different applications can validate their users against different servers.

Each LDAP server definition includes an application name "pattern". When Nirva calls NDAS, NDAS checks the application's name against each server definition's pattern. NDAS will use the first server whose pattern matches the application name. If a single server is to be used for all applications, the pattern "*" (which matches any name) will suffice.

Details

This section describes the fields required by NDAS directory server definitions.

NDAS Server name

Each server definition requires a name. This name is local to NDAS. It need have no relationship to the server's real name.

Description

Each server definition may optionally have a description associated with it. This is free text that provides additional information for administrators.

Match

Each server must have one or more match strings. When Nirva makes a call to NDAS to authenticate a user, NDAS must decide which server to use. It does this by comparing the name of the application that the user is attempting to use against the match strings specified for each server definition.

The order of server definitions is important because NDAS checks each server's match strings in turn. Once it finds a match, it will not check any more.

Each server is allowed multiple match strings. A match string can contain asterisks to indicate; an asterisk will match any group of characters. Following are some examples of application match strings.

Example:

Server	Match strings
`London`	`app1` `app2`
`Paris`	`App*`
`Lane End`	`Testapp`

If application `app1` or `app2` require a user to be authenticated, NDAS will use the `London` server because the application name matches an entry in the `London` server's match string.

For `app3` or `appxxx` or any other application whose name begins with `app`, NDAS will use the `Paris` server because the application name matches an entry in the `Paris` server's match string.

For any application whose name begins with `testapp`, NDAS will use the `Lane End` server.

NDAS will report an error if an application whose name does not begin with `app` or `testapp` makes a call, i.e. whenever it is unable to match an application to a server.

Permissions Processing

NDAS can assign permissions to a session when it successfully authenticates a user.

There are three options, described on the following pages.

There is also a fourth option which is that NDAS never assigns any permissions. This might be useful if an application has its own mechanism for assigning permissions.

Each server can use a different option.

Permissions in a directory field

From the NDAS point of view, the simplest way to assign permissions to a session is to find the permission names in a field in the directory. In this case, each value in the multi-valued field must be have the form *`app`*`:`*`service`*`:`*`permission`* where the colons are coded as shown; two colons must always be present.

- *`app`* is an application name, *`service`* is a service name and *`permission`* is a permission name.
- *`app`* and *`service`* must never appear in the same permission. If neither appears, *`permission`* is assumed to be a system permission. If service is enclosed in parentheses, it designates a web service.

When a user logs onto an application, NDAS ignores permissions for any other application. It also ignores any permissions whose format is incorrect (no error will be raised).

Here is an example of a set of permissions:

```
myapp1::activity1
myapp1::activity2
myapp2::read
myapp2::write
:storage:document_read
:(hello):welcome
::mail_send
```

When a user with the above set of permissions logs onto application `myapp1`, NDAS will assign the user permissions `activity1` and `activity2` for the application, `document_read` for the storage service, `welcome` for the hello web service and system permission `mail_send`.

A directory field contains role names

NDAS allows you to define user "roles". Each role contains a set of permissions for applications, services, web services and the Nirva system.

You can configure NDAS to obtain a list of role names from a field in a logged on user's directory entry. In this case, NDAS assigns the user all the permissions in the NDAS roles specified in the list.

A directory field contains group names

If a field in the user's directory entry contains a list of groups, you can instruct NDAS to map these group names to NDAS role names. In this case, NDAS assigns the user all the permissions in the NDAS roles after it has performed the mapping.

A mapping has the format *`group`*`:`*`role`* where *`group`* is the name of a group obtained from the directory server, the colon is coded as shown and *`role`* is the name of an NDAS role.

Here is an example of a mapping:

```
CN=Developers,OU=MyDomain Users,DC=mydomain,DC=com:Development
CN=Administrators,CN=Builtin,DC=mydomain,DC=com:Administration
```

In this example, a user who is a member of group `CN=Developers,OU=MyDomain Users,DC=mydomain,DC=com` will receive all the permissions from NDAS role `Development` and a user who is a member of group `CN=Administrators,CN=Builtin,DC=mydomain,DC=com` will receive all the permissions from NDAS role `Administration`. A user who is a member of both groups will receive all the permissions from both roles.

Licensing

To use NDAS in a production environment, a license must be purchased for each machine on which it will run.

Without a license, NDAS runs in “demo” mode. This means that once it has received 100 login requests, it will delay all subsequent requests by 10 seconds. In demo mode, the first 100 login requests are actioned the same as in production.

Advanced Usage

NDAS is not restricted to LDAP. It is based on the Java Naming and Directory Interface (JNDI) which allows a program to interface with any directory service. However, to use a directory server other than an LDAP server requires a user supplied Java class. NDAS does not supply this. The name of this class must be specified when an NDAS server is configured.

When the directory server is accessed via LDAP, the supplied Java class `com.sun.jndi.ldap.LdapCtxFactory` should be used.

Reference

Classes

Here are the defined NDAS service classes:

Class	Description
ROLE	Commands pertaining to a single role
ROLES	Commands pertaining to multiple roles
SECURITY	Commands invoked by Nirva when it makes a security call. Applications and other services should not use these commands.
SERVERS	Commands pertaining to multiple server entries
SERVER	Commands pertaining to a single server entry
TEST	Commands for testing the server definitions without the need for an application. These commands can be used to test that the NDAS configuration parameters are correct.

Error codes

The following classes of error can occur:

- ENTRY
- NDAS
- PERMISSION
- SECURITY

ENTRY error class

Value	Description
1	Inserting a role with a duplicate name.

Value	Description
2	Updating the name of an entry so that its name is a duplicate of one already existing.
3	Java runtime exception

NDAS error class

Value	Description
0	No error
101	Required value is missing.
102	Requested item cannot be found.
103	Unexpected error.
104	More than one matching entry was found in the directory - the userid must resolve to a unique name.
105	The application name does not match any NDAS directory server's match string.
106	The role name in a group-role mapping matches multiple roles (should never happen).
107	NDAS has received a request for an unchecked login (login without a password) but the NDAS server does not allow this.
108	NDAS encountered an error when attempting to change the user's password.
109	NDAS encountered an error when encoding the old or new password prior to submitting a change password request.

PERMISSION error class

Value	Description
1	The current user does not have permission to perform the operation.

SECURITY error class

Value	Description
101	The current user does not have permission to perform the operation.
102	Cannot get context (communication problem).
103	Bad password.
110	Bad password syntax.
121	Password has expired.

Permissions

Name	Description
ADMIN	Administration (required to configure the NDAS service)

Commands

For each command, the reference gives the command name, the sources for which the command may be used, the command description, the eventual command permissions, the parameter list and the eventual list of objects created by the command.

The available command sources are:

- Client for all Nirva client interfaces including Nirva client library (nvc).
- Web for commands from a web browser.
- Procedure for commands from a Nirva procedure.
- Service for commands from service to service.
- System for commands from the Nirva system.

ROLE command class

CREATE

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

`ROLE:CREATE` creates a new NDAS role.

Permissions

ADMIN

Parameters

Parameter	Description	Required
NAME	The name of the new NDAS role.	Yes
DESCRIPTION	Free text description of this entry.	No
PERMISSIONS*n*	Specify one PERMISSIONS*n* for each permission in the role. The *n* in PERMISSIONS*n* must be different for each permission. A PERMISSIONS*n* value is a permission with the format *app*:*service*:*permission* where the colons are coded as shown; two colons must always be present. *app* is the name of an application, *service* is the name of a service and *permission* is a permission name. *app* and *service* must never appear in the same permission. If neither appears, the permission is assumed to be a service permission. If *service* is enclosed in parentheses, it is assumed to be a web service.	Yes

---

GET

Source	Use input container	Use output container
Client Web Procedure Service	No	Yes

Description

ROLE:GET creates Nirva objects from an existing NDAS role.

Permissions

ADMIN

Parameters

Parameter	Description	Required
NAME	The name of the NDAS role for which objects are to be created.	Yes

Objects created

String named `role_name` containing the name of the role (the same as was supplied in the `NAME` parameter.

String named `role_description` containing the free text description of the role.

Stringlist named `role_permissions` containing one permission in each entry.

REMOVE

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

`ROLE:REMOVE` removes an existing NDAS role.

Permissions

ADMIN

Parameters

Parameter	Description	Required
`NAME`	The name of the NDAS role to be removed	Yes

Objects created

None

UPDATE

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

ROLE:UPDATE updates a new NDAS role.

Permissions

ADMIN

Parameters

Parameter	Description	Required
NAME	The name of the NDAS role to be updated.	Yes
DESCRIPTION	Free text description of this entry.	No
PERMISSION*n*	Specify one PERMISSION*n* for each permission in the role. The *n* in PERMISSION*n* must be different for each permission. A PERMISSION*n* value is a permission with the format *app*:*service*:*permission* where the colons are coded as shown; two colons must always be present. *app* is the name of an application, *service* is the name of a service and permission is a permission name. *app* and *service* must never appear in the same permission. If neither appears, the permissions is assumed to be a service permission. If *service* is enclosed in parentheses, it is assumed to be a web service.	Yes

ROLES command class

LIST

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

ROLES:LIST creates a table that lists all the defined NDAS roles.

Permissions

ADMIN

Parameters

None

Objects created

Table named `roles`. Each row in the table describes an NDAS role.

Table column name	Description
`Name`	The name of the role.
`Description`	A free text description of the role.
`Permission`	A list of permissions in the format *`app:service:permission`* where *`app`* is the name of an application and service is the name of a service (or a web service if it is in parentheses. No permission will contain both an application name and a service name. A service permission will be recognised by containing neither an application name nor a service name.

SECURITY command class

LOGIN

Source	Use input container	Use output container
Client Web Procedure Service	No	Yes

Description

`SECURITY:LOGIN` is called automatically by Nirva during user login for an application that has `(NDAS)` specified as its Application for security. It is documented here for completeness. Applications and external services should never call it.

When `CONTROL_PASSWORD` is set to `NO`, NDAS queries the directory server but does not validate the password. This allows Nirva to create a session for a user when running a scheduled task.

Permissions

None

Parameters

Parameter	Description	Required
APPLICATION	The name of the application which the user is logging on to.	Yes
USER	The user id of the user logging on.	Yes
PASSWORD	The password to be used for authentication. This is not used if CONTROL_PASSWORD has the value NO.	Yes
CONTROL_PASSWORD	YES if the password should be validated; otherwise NO.	No

Objects created

String named user_full_name containing the user's full name obtained from the directory server.

SERVER command class

CREATE

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

SERVER:CREATE defines a new NDAS directory server.

Permissions

ADMIN

Parameters

Parameter	Description	Required
SERVER_NAME	The name for NDAS of the new directory server entry.	Yes
MATCH	One or more application name patterns.	Yes
DESCRIPTION	Free text description of this entry.	No
USERIDFIELDNAME	Name of the field in the directory server that contains user	Yes

<table>
<tr><th>Parameter</th><th>Description</th><th>Required</th></tr>
<tr><td></td><td>names.</td><td></td></tr>
<tr><td>COMPOSITENAME</td><td>Root namespace name in which NDAS will search for entries.</td><td>Yes</td></tr>
<tr><td>CONTEXTFACTORY</td><td>Name of the Java class that provides the interface between NDAS and the directory server.</td><td>Yes</td></tr>
<tr><td>PROVIDERURL</td><td>The URL that NDAS will use to connect to the directory server.</td><td>Yes</td></tr>
<tr><td>PERMISSIONSMETHOD</td><td>1 if no permissions processing is required. In this case, the contents of GROUPNAMESFIELDNAME, GROUPROLEMAPPINGMETHOD and GROUPROLEMAPPINGS are not required and will not be used (though they will be saved in the server entry). When this option is used, NDAS will not assign a logged on user any permissions though the application might assign its own permissions to the user independently of NDAS.

2 if a list of roles or groups for the user is stored in a specific server field whose name is specified by the GROUPNAMESFIELDNAME parameter. NDAS uses the GROUPROLEMAPPINGMETHOD value to know whether the GROUPNAMESFIELDNAME field contains role names or group names.

3 if NDAS will find a list of permissions in a specific server field whose name is specified by the PERMISSIONSFIELDNAME parameter.

If no value or an invalid value is supplied, NDAS will set the value to 1.</td><td>No</td></tr>
<tr><td>NAMESFIELDNAME</td><td>The name of a server field that contains a list of names. NDAS determines how to use these names based upon the value of PERMISSIONSMETHOD. The following table summarises the possibilities.

<table><tr><th>PERMISSIONSMETHOD</th><th>NAMESFIELDNAME</th></tr><tr><td>1</td><td>No meaning (not used).</td></tr><tr><td>2</td><td>A list of roles (if GROUPROLEMAPPINGMETHOD is 1) or groups (if GROUPROLEMAPPINGMETHOD is 2)</td></tr><tr><td>3</td><td>A list of permissions.</td></tr></table></td><td>Yes</td></tr>
<tr><td>GROUPROLEMAPPINGMETHOD</td><td>1 if each name retrieved from the NAMESFIELDNAME field represents a role name. In this case the contents of the GROUPROLEMAPPINGS parameter is not used.

2 if each group name must be mapped by NDAS. In this case, the GROUPROLEMAPPINGS parameter specifies the mappings.</td><td>No</td></tr>
</table>

Parameter	Description	Required
	If no value or an invalid value is supplied, NDAS will set the value to `1`.	
GROUPROLEMAPPINGS	A list of mappings in the form *group*:*role* where *group* is the name of a group retrieved from the directory server and *role* is the name of a role to be used by the application. Separate the mappings with new lines.	Yes
USERFULLNAMEFIELDNAME	The name of a server field that contains a user's full name.	Yes
SERVERSOFTWARE	`1` if the directory server is a generic LDAP server. `2` if the directory server is Microsoft Active Directory (`1` can be used for Microsoft Active Directory but the configuration is less flexible). If not specified, a default of `1` is used.	No
ACTIVEDIRECTORYDOMAINNAME	This parameter is used only when `SERVERSOFTWARE` is set to `2`. Its value should be the name of the Active Directory domain. If not specified for Active Directory, users must enter their userid as domain\userid but if a value is specified here, they need enter only userid.	No
CONNECTIONTIMEOUT	An integer value that defines the number of seconds within which the server must respond. If it does not respond within this time, NDAS reports an error. If not specified, a default of `20` is used.	No
ALLOWNIRVALOGIN	If this value is `true`, NDAS will allow Nirva to request a login of a user without needing to supply password. If it has any other value, this will not be allowed.	No
NIRVALOGINQUERYANONYMOUSLY	If this value is `true`, NDAS will query the directory server without first attempting to identify itself. This will only work if the server allows queries from anonymous users. Any value other than `true` is interpreted as `false`. If not specified, a default of `false` is used.	
NIRVALOGINUSER	A string value that specifies the user that NDAS will use to identify itself to the directory server when logging in a user in response to a request from Nirva to login the user without a password. This is not used when `NIRVALOGINQUERYANONYMOUSLY` has the value true.	
NIRVALOGINPASSWORD	A string value that is the password for the user specified by `NIRVALOGINUSER`.	

Parameter	Description	Required
CHANNELSECURITY	The security mechanism that NDAS uses to communicate with the LDAP server: 1 if none. 2 if SSL.	
KEYSTORE	The full path name to the certificate store, e.g. C:/nirva/Java/lib/security/cacerts. This is ignored if CHANNELSECURITY is not set to 2 (SSL).	

GET

Source	Use input container	Use output container
Client Web Procedure Service	No	Yes

Description

SERVER:GET creates Nirva objects from an existing NDAS server.

Permissions

ADMIN

Parameters

Parameter	Description	Required
SERVER_ID	The internal id of the NDAS server for which objects are to be created.	Yes

Objects created

String named server_id containing the internal id of the server (the same as was supplied in the SERVER_ID parameter.

String named server_name containing the name of the server.

Stringlist named server_match containing one entry for each for the server's match strings.

String named server_description containing the free text description of the server.

String named `server_useridfieldname` containing the name of the directory server field where the userid is stored.

String named `server_compositename` containing the server's composite name value.

String named `server_contextfactory` containing the name of the Java class that provides the interface to the directory server.

String named `server_providerurl` containing the url of the directory server.

Integer named `server_permissionsmethod` containing `1`, `2` or `3` indicating how permissions are to be handled for this server.

String named `server_namesfieldname` containing the name of the directory server field that contains group or role names.

Integer named `server_grouprolemappingmethod` containing `1` (if server_namesfieldname contains NDAS role names) or `2` (if `server_namesfieldname` contains group names to be mapped with the `server_grouprolemappings` mappings).

Stringlist named `server_grouprolemappings` containing mappings from group names to role names.

String named `server_userfullnamefieldname` containing the name of the directory server field that contains a user's full name.

Integer named `server_serversoftware` containing `1` if the server is a generic LDAP server or `2` if the server is Microsoft Active Directory.

String named `server_activedirectorydomainname` containing the name of the Microsoft Active Directory server domain name.

Integer named `server_connectiontimeout` containing the timeout value.

Boolean named `server_allownirvalogin`. The value is `true` if NDAS will accept a request from Nirva to login a user without supplying a password; otherwise `false`.

Boolean named `server_nirvaloginqueryanonymously`. The value is `true` if NDAS should query the directory server anonymously when a Nirva makes a user login request without supplying a password; otherwise `false`.

String named `server_nirvaloginuser` containing the name of the user that NDAS will use to identify itself to the directory server when querying for information about a user for whom Nirva has supplied no password.

String named `server_nirvaloginpassword` containing the password for the user specified by `server_nirvaloginuser`.

Integer name `server_channelsecurity` indicating the security mechanism that NDAS uses to communicate with the server: `1` (if none) or `2` (if SSL).

MOVE_DOWN

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

`SERVER:MOVE_DOWN` moves the specified server down in the order of servers. The order of servers is important because NDAS checks each server's match string(s) in turn. When it finds a match, it does not check any more servers.

Permissions

ADMIN

Parameters

Parameter	Description	Required
`SERVER_ID`	The internal id of an NDAS server. This server and the one below it will swap places. If the server is the last server, nothing happens though no error is reported.	Yes

Objects created

None.

MOVE_UP

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

`SERVER:MOVE_UP` moves the specified server up in the order of servers. The order of servers is important because NDAS checks each server's match string(s) in turn. When it finds a match, it does not check any more servers.

Permissions

ADMIN

Parameters

Parameter	Description	Required
SERVER_ID	The internal id of an NDAS server. This server and the one above it will swap places. If the server is the first server, nothing happens though no error is reported.	Yes

Objects created

None.

REMOVE

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

SERVER:REMOVE removes an existing NDAS server.

Permissions

ADMIN

Parameters

Parameter	Description	Required
SERVER_ID	The internal id of the server to be removed	Yes

Objects created

None

UPDATE

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

`SERVER:UPDATE` updates an existing NDAS directory server.

Permissions

ADMIN

Parameters

Parameter	Description	Required
`SERVER_ID`	The internal NDAS id of the directory server to be updated.	Yes
`NAME`	The name for NDAS of the directory server entry.	Yes
`MATCH`	One or more application name patterns.	Yes
`DESCRIPTION`	Free text description of this entry.	No
`USERIDFIELDNAME`	Name of the field in the directory server that contains user names.	Yes
`COMPOSITENAME`	Root namespace name in which NDAS will search for entries.	Yes
`CONTEXTFACTORY`	Name of the Java class that provides the interface between NDAS and the directory server.	Yes
`PROVIDERURL`	The URL that NDAS will use to connect to the directory server.	Yes

<table>
<tr><th>Parameter</th><th>Description</th><th>Required</th></tr>
<tr><td>PERMISSIONSMETHOD</td><td>1 if no permissions processing is required. In this case, the contents of GROUPNAMESFIELDNAME, GROUPROLEMAPPINGMETHOD and GROUPROLEMAPPINGS are not required and will not be used (though they will be saved in the server entry). When this option is used, NDAS will not assign a logged on user any permissions though the application might assign its own permissions to the user independently of NDAS.

2 if a list of roles or groups for the user is stored in a specific server field whose name is specified by the GROUPNAMESFIELDNAME parameter. NDAS uses the GROUPROLEMAPPINGMETHOD value to know whether the GROUPNAMESFIELDNAME field contains role names or group names.

3 if NDAS will find a list of permissions in a specific server field whose name is specified by the PERMISSIONSFIELDNAME parameter.

If no value or an invalid value is supplied, NDAS will set the value to 1.</td><td>No</td></tr>
<tr><td>NAMESFIELDNAME</td><td>The name of a server field that contains a list of names. NDAS determines how to use these names based upon the value of PERMISSIONSMETHOD. The following table summarises the possibilities.

<table><tr><th>PERMISSIONSMETHOD</th><th>NAMESFIELDNAME</th></tr><tr><td>1</td><td>No meaning (not used).</td></tr><tr><td>2</td><td>A list of roles (if GROUPROLEMAPPINGMETHOD is 1) or groups (if GROUPROLEMAPPINGMETHOD is 2)</td></tr><tr><td>3</td><td>A list of permissions.</td></tr></table></td><td>Yes</td></tr>
<tr><td>GROUPROLEMAPPINGMETHOD</td><td>1 if each name retrieved from the NAMESFIELDNAME field represents a role name. In this case the contents of the GROUPROLEMAPPINGS parameter is not used.

2 if each group name must be mapped by NDAS. In this case, the GROUPROLEMAPPINGS parameter specifies the mappings.

If no value or an invalid value is supplied, NDAS will set the value to 1.</td><td>No</td></tr>
</table>

Parameter	Description	Required
GROUPROLEMAPPINGS	A list of mappings in the form *group*:*role* where *group* is the name of group retrieved from the directory server and *role* is the name of a role to be used by the application. Separate the mappings with new lines.	No
USERFULLNAMEFIELDNAME	The name of a server field that contains a user's full name.	Yes
SERVERSOFTWARE	1 if the directory server is a generic LDAP server. 2 if the directory server is Microsoft Active Directory (1 can be used for Microsoft Active Directory but the configuration is less flexible). If not specified, a default of 1 will be used.	No
ACTIVEDIRECTORYDOMAINNAME	This parameter is used only when SERVERSOFTWARE is set to 2. Its value should be the name of the Active Directory domain. If not specified for Active Directory, users must enter their userid as domain\userid but if a value is specified here, they need enter only userid.	No
CONNECTIONTIMEOUT	An integer value that defines the number of seconds within which the server must respond. If it does not respond within this time, NDAS reports an error. If this value is not specified, a default of 20 is used.	No
ALLOWNIRVALOGIN	If this value is true, NDAS will allow Nirva to request a login of a user without needing to supply password. If it has any other value, this will not be allowed.	No
NIRVALOGINQUERYANONYMOUSLY	If this value is true, NDAS will query the directory server without first attempting to identify itself. This will only work if the server allows queries from anonymous users.	No
NIRVALOGINUSER	A string value that specifies the user that NDAS will use to identify itself to the directory server when logging in a user in response to a request from Nirva to login the user without a password. This is not used when NIRVALOGINQUERYANONYMOUSLY has the value true.	No
NIRVALOGINPASSWORD	A string value that is the password for the user specified by NIRVALOGINUSER.	No
CHANNELSECURITY	The security mechanism that NDAS uses to communicate with the LDAP server: 1 if none (default) 2 if SSL	No

Parameter	Description	Required
KEYSTORE	The full path name to the certificate store, e.g. C:/nirva/Java/lib/security/cacerts. This is ignored if CHANNELSECURITY is not set to (SSL).	

SERVERS command class

LIST

Source	Use input container	Use output container
Client Web Procedure Service	No	Yes

Description

SERVERS:LIST creates a table that lists all the defined NDAS directory servers.

Permissions

ADMIN

Parameters

None

Objects created

Table named servers with columns as shown below.

Table column name	Description
Server_id	The unique id of the server.
Name	The name of the server.
Match	Each cell line is an application name match for this server.
Description	The free text description for this server
User_field_name	The name of the field defined in the directory server that contains user names.

Table column name	Description
Namespace	The composite name that is used as the basis of each directory server search.
Java_context_factory	The name of the Java class that provides the interface between NDAS and the directory server.
Provider_URL	The URL of the directory server.
Permissions_method	1 if no permissions processing is required. In this case, the contents of the following three columns are meaningless. 2 if a server field contains group or role names. In this case, Names_field_name contains the name of the field and Group_role_mapping_method determines whether the names in the field are group names or role names. 3 if a server field contains permissions.
Names_field_name	The name of a server field that contains a list of groups, roles or permissions, depending upon the settings of Permissions_method and Group_role_mapping_method.
Group_role_mapping_method	1 if each name in the field specified by Names_field_name is a role name. In this case the contents of Group_role_mappings are meaningless. 2 if each name in the field specified by Names_field_name is a group name. In this case, these names will be mapped to NDAS roles by means of the mappings in the Group_role_mappings field.
Group_role_mappings	Each cell line is a mapping in the form group:role where group is the name of group retrieved from the directory server and role is the name of a role to be used by the application.
User_full_name	The name of the field in the directory where NDAS can find the user's full name.
Server_software	1 if the server is a generic LDAP server. 2 if the server is a Microsoft Active Directory server.
Active_Directory_domain_name	The name of the Microsoft Active Directory domain. This is used only when Server_software is 1.
Connection_timeout	An integer indicating the number of seconds that NDAS will wait for a response from the directory server before reporting an error.
Allow_unchecked_login	true if NDAS is allowed to login a user when Nirva request a login without a password. Nirva makes this request when running a scheduled task; otherwise false.

Table column name	Description
Nirva_login_Query_anonymously	true if NDAS will query the server anonymously when obtaining information for a user that Nirva has requested be logged on without a password; otherwise false.
User_for_login_without_password_query	The userid that NDAS will use when querying the directory server for a user without a password. This is used only when Nirva_login_Query_anonymously has the value true.
Password_for_login_without_password_query	The password for the userid specified by User_for_login_without_password_query.
Channel_security	An integer indicating the security mechanism that NDAS uses to communicate with the server: 1 for none. 2 for SSL.
Keystore	The full path to the file containing the certificate store for use with SSL.

TEST command class

MATCH

Source	Use input container	Use output container
Client Web Procedure Service	No	Yes

Description

TEST:MATCH returns the name and id of the directory server that will used to authenticate users of a specific application.

Permissions

ADMIN

Parameters

Parameter	Description	Required

Parameter	Description	Required
APPNAME	The name of the application for which the directory server is to be determined.	Yes

Objects created

String object named match_server_name containing the name of the entry that the application name matches.

CONFIGURATION

Source	Use input container	Use output container
Client Web Procedure Service	No	No

Description

TEST:CONFIGURATION sends a name and password to a directory server and reports the result. This is useful for confirming that a directory server has been configured correctly.

When testing a configuration, the server's match string is not used. This command is provided to allow a connectivity test from Nirva to the directory server.

Permissions

ADMIN

Parameters

Parameter	Description	Required
CONTEXTFACTORY	The name of the Java class providing the interface to the directory server.	Yes
PROVIDERURL	The URL of the directory server.	Yes
USERFIELDNAME	The name of the field containing the user's login id.	Yes
COMPOSITENAME	Name of the field in the directory server that contains user names.	Yes
NDAS_USER	The user to be used in the request to the directory server.	Yes
NDAS_PASSWORD	The password for the user to be used in the request to the directory server.	Yes

Objects created

???

Appendix A : Configuring SSL for Microsoft Active Directory

Introduction

If the directory server is Microsoft Active Directory and you want users to be able to change their passwords via NDAS, you must configure NDAS to use SSL for its communication. Microsoft Active Directory does not allow passwords to be changed via LDAP unless SSL is used.

This Appendix describes how to configure SSL for NDAS so that can communicate with Microsoft Active Directory using SSL. It assumes that Microsoft Active Directory is running on Microsoft Windows Server 2003 and that you are running Nirva on a Windows machine though the process will be very similar for Nirva on a Unix machine.

Overview

There are three steps to configuring NDAS to use SSL.

1. Obtain a digital certificate from the Active Directory. This will identify NDAS to Active Directory.
2. Save the certificate in the Java Certificate Store. When Nirva is installed, it has a certificate store that can be used (recommended). Alternatively, you can create a new certificate store.
3. Configure the NDAS server entry to use SSL and the certificate store.

Obtaining the digital certificate

On the machine where Nirva is running, open Internet Explorer (not Firefox or any other browser) and go to http://server/certsrv where server is the name of your Active Directory server.

Click Download a CA certificate, certificate chain, or CRL. The "Download a CA Certificate, Certificate Chain, or CRL" page will be displayed.

Leave "Encoding method" as DER. Click Download CA certificate. A download will begin. Save the file as `certnew.cer` (or any other `.cer` name of your choice) in a location of your choice.

You have now obtained your digital certificate.

Adding the certificate to the store

The following instructions assume that Nirva has been installed in C:\Nirva. If this is not the case, change C:\Nirva appropriately wherever it occurs.

Open a command window.

Navigate to C:\nirva\Java\bin.

Enter the following command:

```
keytool -import -alias mydomain -file "c:\temp\certnew.cer" -keystore
"C:\nirva\Java\lib\security\cacerts"
```

Replace `mydomain` with the name of your domain and `c:\temp\certnew.cer` with the location of the certificate that you previously downloaded. The command above will save the certificate in the certificate store supplied with Nirva, i.e. `C:\nirva\Java\lib\security\cacerts`. You can change this if you wish. If you specify a certificate store that does not exist, it will be automatically created.

When you enter the keytool, you wil be prompted for a password. If you are creating a new keystore enter any password and remember it for future updates. NDAS does not need to know the password.

Configure the NDAS server entry for SSL

Open the configuration page for the NDAS server that you wish to modify.

Click the Secure channel tab.

Select the SSL radio button.

In the Keystore location field, enter the name of the certificate store where you saved the digital certificate.

Click the Server details tab.

Change the port for the server from 389 to 636 (these are the ports conventionally used by LDAP: 389 for non-SSL, 636 for SSL; your server might be different).