LDAP Functions

LDAP (Lightweight Directory Access Protocol) is a protocol that allows you to search and update a directory. It is supported by numerous directory servers such as Microsoft Active Directory and Netscape Directory Server. A directory usually contains information about users and network resources, but can contain any kind of data.These functions allow scripts and transformations to connect to a directory using LDAP.

The use pattern follows these steps:

- **`LDAPConnect()`:** Establishes a connection to the server
- **`LDAPAdd()`**, **`LDAPReplace()`**, **`LDAPRemove()`**,..**:** Queues actions to be taken
- **`LDAPExecute()`:** Executes the queue on the server

For searching, you can follow this pattern:

- **`LDAPConnect()`:** Establishes a connection to the server
- **`LDAPSearch()`:** Executes the search on the server and returns the result

Setting LDAP Passwords Programmatically

As of Harmony version 9.9, LDAP passwords can be set using these LDAP functions. The password needs to first be converted to binary before it can be set. This script fragment shows how this could be accomplished:

```
LDAPConnect("directory.company.example.com:10389", "admin",
$adminpassword, 0);
$plain = "J1tterb1t!@#";
$pwd = StringToHex($plain);
$i=0;
$newPwd = "";
While($i < Length($pwd)/2,
    $newPwd = $newPwd + Mid($pwd, $i * 2, 2) + "00";
    $i = $i+1;
);
$newPwd = HexToBinary($newPwd);

LDAPReplace("unicodePwd", $newPwd);
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

ArrayToMultipleValues

Declaration

```
string ArrayToMultipleValues(array arr)
```

Syntax

```
ArrayToMultipleValues(<arr>)
```

Required Parameters

- **`arr`:** An array with values to be used to populate a multiple-valued LDAP attribute

Description

Signals that an array should be used to populate a multiple-valued LDAP attribute when mapping to an LDAP target. The array is converted to XML and interpreted when the LDAP server is written to.

Examples

On This Page

- ArrayToMultipleValues
- LDAPAdd
- LDAPConnect
- LDAPDeleteEntry
- LDAPExecute
- LDAPRemove
- LDAPRename
- LDAPReplace
- LDAPSearch

Search in This Topic

Related Topics

- Cloud Studio
- Functions
- Scripts

Last updated: Feb 12, 2020

```
// Create two LDAP attributes using an array
arr = Array();

// Length(arr) will always return the index of
// the next-after-last element of an array
arr[Length(arr)] = "First instance";
arr[Length(arr)] = "Second instance";

// The array will be converted to the string
// "<JB_Array><it>First instance</it><it>Second instance</it></JB_Array>"
result = ArrayToMultipleValues(arr);
```

[Back to Top]

LDAPAdd

Declaration

```
bool LDAPAdd(string ldapType, string ldapValue)
```

Syntax

```
LDAPAdd(<ldapType>, <ldapValue>)
```

Required Parameters

- **`ldapType`:** The LDAP record or attribute type
- **`ldapValue`:** The value to be added for the type

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

After a connection is established, the `LDAPAdd()` function is used to add entries and attributes to the connected LDAP directory. The value is added to the node specified in the `LDAPExecute()` function, which should be called immediately afterwards to have the changes take effect.

See also the `LDAPConnect()` and `LDAPExecute()` functions.

Examples

```
// Adding a user entry to a directory:
LDAPAdd("objectClass", "user");
LDAPAdd("cn", "wright");
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");

// Adding attributes to a user:
LDAPAdd("description", "Thinks a lot.");
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

[Back to Top]

LDAPConnect

Declaration

```
bool LDAPConnect(string hostname, string user, string password[, int mode,
long port])
```

Syntax

```
LDAPConnect(<hostname>, <user>, <password>[, <mode>, <port>])
```

Required Parameters

- **`hostname:`** Hostname of an LDAP server to connect to; can include a port, such as `directory.example.com:10389`
- **`user:`** User name to connect to the LDAP host
- **`password:`** Password to use to connect to the LDAP host

Optional Parameters

- **`mode:`** Connection mode; one of:
  - 0: Non-secure connection; default port: 389
  - 1: Secure connection; default port: 389
  - 2: Secure connection using LDAP over SSL; default port: 636
- **`port:`** Port to use to connect to the LDAP host (`-1` indicates to use the default LDAP ports)

Description

Connects to a directory using LDAP.

See also the `LDAPExecute()` function.

TIP: When using this command, use a Jitterbit project variable to store the password in a secure manner that will not be visible when the script is being viewed.

Examples

```
// Connects to an LDAP server using hostname:port, username, password, mode
connected = LDAPConnect("directory.company.example.com:10389", "admin",
$adminpassword, 0);
```

[Back to Top]

LDAPDeleteEntry

Declaration

```
bool LDAPDeleteEntry(string distinguishedName)
```

Syntax

```
LDAPDeleteEntry(<distinguishedName>)
```

Required Parameters

- **`distinguishedName:`** An LDAP distinguished name to be removed from the currently connected server

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

After a connection is established, the `LDAPDeleteEntry()` function is used to remove an entry specified by a distinguished name.

See also the `LDAPConnect()` and `LDAPExecute()` functions.

Examples

```
LDAPDeleteEntry("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

[Back to Top]

LDAPExecute

Declaration

```
bool LDAPExecute(string distinguishedName)
```

Syntax

```
LDAPExecute(<distinguishedName>)
```

Required parameters:

- **`distinguishedName`:** An LDAP distinguished name on the currently connected server

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

After a connection is established, the `LDAPExecute()` function is used to execute one or more modifications (add, remove, replace) that have previously been specified with the `LDAPAdd()`, `LDAPRemove()`, or `LDAPReplace()` functions.

See also the `LDAPConnect()` and `LDAPExecute()` functions.

Examples

```
// Adding LDAP entries and executing them
LDAPAdd("description", "Thinks a lot.");
LDAPReplace("telephoneNumber", "(510) 555 1000");
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

[Back to Top]

LDAPRemove

Declaration

```
bool LDAPRemove(string ldapType, string ldapValue)
```

Syntax

```
LDAPRemove(<ldapType>, <ldapValue>)
```

Required Parameters

- **`ldapTyp`:** The LDAP attribute type
- **`ldapValue`:** The value to be removed for the specified attribute type

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

Once a connection is established, the `LDAPRemove()` function is used to remove an attribute of a specified type and with a specified value.

If the attribute type of that value is not found, an error is thrown. See also the `LDAPConnect()` and `LDAPExecute()` functions.

Examples

```
// Removing an LDAP entry and executing it
LDAPRemove("telephoneNumber", "(510) 555-1000");
LDAPAdd("telephoneNumber", "(510) 555-2000");
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

[Back to Top]

LDAPRename

Declaration

```
bool LDAPRename(string distinguishedName, string newRDN[, string
newParent, bool deleteOldRDN])
```

Syntax

```
LDAPRename(<distinguishedName>, <newRDN>[, <newParent>, <deleteOldRDN>])
```

Required Parameters

- **`distinguishedName:`** The path of the directory entry (distinguished name) to be renamed
- **`newRDN:`** The new relative distinguished name

Optional Parameters

- **`newParent:`** The distinguished name of the new parent of this entry; the default is empty
- **`deleteOldRDN:`** If `true`, the old relative distinguished name will be deleted; the default is `false`

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

Once a connection is established, the `LDAPRename()` function is used to change the distinguished name of an entry in the connected directory.

See also the `LDAPConnect()` and `LDAPExecute()` functions.

Examples

```
LDAPRename("telephoneNumber", "telephoneNumberHome");
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

[Back to Top]

LDAPReplace

Declaration

```
bool LDAPReplace(string ldapType, string ldapValue)
```

Syntax

```
LDAPReplace(<ldapType>, <ldapValue>)
```

Required Parameters

- **`ldapType`:** The LDAP record or attribute type
- **`ldapValue`:** The new value to be set for the type

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

Once a connection is established, the `LDAPReplace()` function is used to replace an existing attribute's value with a new value.

See also the `LDAPConnect()` and `LDAPExecute()` functions.

> **NOTE:** If you use `LDAPReplace()` to replace an existing attribute's value with a new value in an Active Directory, the text may be converted to a different result than the entry. For example, *François* would return *FranÃ§Ois*. For more information on adding UTF-8 support, see the LDAP entry in Editing the Configuration File - jitterbit.conf.

Examples

```
// Replacing an LDAP entry and executing it
LDAPReplace("telephoneNumber", "(510) 555-1000");
LDAPExecute("CN=wright,CN=Users,DC=company,DC=example,DC=com");
```

[Back to Top]

LDAPSearch

Declaration

```
string LDAPSearch(string path, string filter, int detail, string attribute1
[, ... string attributeN])
```

Syntax

```
LDAPSearch(<path>, <filter>, <detail>, <attribute1>[, ... <attributeN>])
```

Required Parameters

- **`path`:** The distinguished name used as the base of the search
- **`filter`:** A query string search filter, as defined by RFC 4515
- **`detail`:** Details to be returned:
    - **`0`:** Pass to return a simple string that is the first attribute found that matches the filter
    - **`1`:** Pass to return an XML representation of the search results. This can be set to a data element using the `Set()` function and then accessed later using `Get()` and an XPath query using the `SelectSingleNode()` function.
    - **`2`:** Pass to return a 2-dimensional array, where each row represents an LDAP entry and where the row elements are the values of the attributes. Access the elements using the `Get()` function.

- **`attribute,... attributeN:`** An attribute to be included in the search result (in other words, the attribute you are searching for). Additional attributes can be specified; separate them by commas.

Description

To use this function, the `LDAPConnect()` function must first be used to establish a connection with the LDAP directory.

Once a connection is established, the `LDAPSearch()` function is used to search within the connected directory.

See also the `LDAPConnect()` and `LDAPExecute()` functions.

Controlling Jitterbit Variables

These Jitterbit variables affect the search:

Variable	Value (s)	Description
`jitterbit.scripting.ldap.scope`	0	Limit the scope of the query to the base-entry only
	1	Search all entries in the first level below the base-entry, excluding the base-entry (default)
	2	Search the base-entry and all entries in the tree below the base
`jitterbit.scripting.ldap.include_dn_in_results`	`true`	Include the distinguished name in the search result (default: `false`)
`jitterbit.scripting.ldap.use_paged_search`	`true`	Use a paged search; useful for retrieving large result sets (default: `false`)
`jitterbit.scripting.ldap.max_search_results`	*n*	Limit the number of search results to *n* results; a default is usually set by the LDAP server (default unlimited: `-1`)
`jitterbit.scripting.ldap.return_null_if_no_results`	`true`	Make LDAPSearch() return null if the search does not return any results (default: false)

Examples

```
// Connecting to an LDAP server using a password set
// in a global variable ("$ldapPassword") and then performing a search
LDAPConnect("directory.company.example.com", "ghvwright", $ldapPassword,
0);
directorySearchResults = LDAPSearch("CN=Users,DC=company,DC=example,
DC=com",
  "(&(objectCategory=person)(objectClass=user))", 1, "name", "whenCreated",
  "description", "telephoneNumber");
```

[Back to Top]