# LDAPasp_RFC4511 Test Port for TTCN-3 Toolset with TITAN, User Guide

Szabolcs Béres

Version 198 17-CNL 113 513, Rev. D, 2012-10-12

# Table of Contents

# About This Document

## How to Read This Document

This is the User Guide for the LDAPasp_RFC4511 test port. The LDAP test port is developed for the TTCN-3 Toolset. This document should be read together with Function Specification [4].

## Presumed Knowledge

The knowledge of the TITAN TTCN-3 Test Executor [3] and the TTCN-3 language [1] is essential. Basic knowledge of the LDAP and SSL protocols is valuable when reading this document.

# System Requirements

In order to operate the LDAP test port the following system requirements must be satisfied:

- Platform: any platform supported by TITAN RTE and OpenSSL
- TITAN TTCN-3 Test Executor version R8A (1.8.pl0) or higher installed. For installation guide see [2].

| NOTE | This version of the protocol module is not compatible with TITAN releases earlier than R8A. |
|---|---|

- The Abstract_Socket CNL 113 384, rev. R6A or later product has to be installed.

If SSL is used, the same OpenSSL must be installed as used in TITAN. For installation guide see [5].

# Fundamental Concepts

The test port establishes connection between the TTCN-3 test executor and SUT and transmits/receives messages. The transport channel can be TCP or SSL.

When used with connection ASPs, the test port provides ASPs to let the TTCN-3 code control the creation and closing of TCP connections or the open and close of TCP server listening port.

# The Test Port

# Overview

The LDAPasp_RFC4511 test port offers LDAP message primitives to the test suite in TTCN-3 format. The TTCN-3 definition of the LDAP messages can be found in a separate ASN.1 module. This module should be imported into the test suite. For more information on LDAP see [6].

The LDAP test port also offers LDIF importation, population, depopulation functionality. For more information on LDIF see [8].

# Sending/Receiving LDAP Messages

LDAP message can be sent/received with the following ASP:

## ASP_LDAP_msg

- usage:

  Send to the test port to send an LDAP message. The test port will send this ASP in case of incoming LDAP message.

- fields:

  `client_id` integer optional – used to identify the connection. In case of the sending operation the values get from the `ASP_LDAP_connect_result` or `ASP_LDAP_connected` ASPs can be used. In case of client operation, if only one connection is alive, this value can be omitted.

# Connection ASPs

When using connection ASPs, i.e. the `use_connection_ASPs`, Test Port parameter is set to the value `"yes"`, the following ASPs are used by the test port:

## ASP_LDAP_connect

- usage:

  Send to the test port to connect to a remote host.

- fields:
  - `hostname` charstring – remote host name to connect to
  - `portnumber` integer – remote port to connect to
  - `local_hostname` charstring optional – local host interface to connect from. If used with the `"omit"` value, the Test Port will use the value set in the `local_hostname` Test Port Parameter or the `"localhost"` value.
  - `local_portnumber` integer optional – local port number to connect from. If used with the `"omit"` value, the Test Port will use the value set in the `local_port` Test Port Parameter or a random port.

## ASP_LDAP_connect_result

- usage:

  The Test Port sends it to the test suite as a result of the `ASP_LDAP_connect` ASP if the

use_connection_ASPs Test Port parameter is set to the value "yes".

- fields:

  client_id integer – contains -1 in case of connection error, otherwise the id of the connection which can be used during the send operation in the ASP_LDAP_msg ASP.

## ASP_LDAP_connected

- usage:

  The Test Port sends it to the test suite if an incoming connection is accepted and if the use_connection_ASPs Test Port parameter is set to the value "yes".

- fields:
  - client_address charstring – host name of the connected client
  - client_id integer – the ID of the connected client that can be used during the send operation in the ASP_LDAP_msg ASP.

## ASP_LDAP_listen

- usage:

  Send to the Test Port to open a server listening port. If a listening port was already opened, it will be closed before opening a new one. The client connections will remain alive.

- fields:
- portnumber integer optional - the port number on the local host to listen on. If used with the "omit" value, the Test Port will use the value set in the local_port Test Port Parameter or a random port.
- local_hostname charstring optional – the host name on the local host to listen. If used with the "omit" value, the Test Port will use the value set in the local_hostname Test Port Parameter or the "localhost" value.

## ASP_LDAP_listen_result

- usage:

  The Test Port sends it to the test suite as a result of the ASP_LDAP_listen ASP if the use_connection_ASPs Test Port parameter is set to the value "yes".

- fields:

  portnumber integer – contains -1 in case of listen error, otherwise the port on which the server listens on

### ASP_LDAP_close

- usage:

  Send to the Test Port to close a specific or all connections.

- fields:

  `client_id` integer optional – specify the connection to close, or if omitted, it means all alive connections

### ASP_LDAP_closed

- usage:

  The Test Port sends it to the test suite if a connection is closed by the other side and if the if the `use_connection_ASPs` Test Port parameter is set to the value `"yes"`.

- fields:

  `client_id` integer optional – identifies the closed connection

### ASP_LDAP_shutdown

- usage:

  Send to the Test Port to shut down a server listening port

- fields:

  None.

# Installation

Since the LDAPasp_RFC4511 test port is used as a part of the TTCN-3 test environment this requires TTCN-3 Test Executor to be installed before any operation of the LDAP test port. For more details on the installation of TTCN-3 Test Executor see the relevant section of [2].

The compilation of SSL related code parts can be disabled by not defining the `AS_USE_SSL` macro in the *Makefile* during the compilation.

When building the executable test suite the libraries compiled for the OpenSSL toolkit (if the `AS_USE_SSL` macro is defined) should also be linked into the executable along with the TTCN-3 Test Executor, i.e. the OpenSSL libraries should be added to the *Makefile* generated by the TITAN executor (see example in Makefile. To compile the source files you will also need the OpenSSL developer toolkit which contains the header files used by the source. If Share Objects (*.so*) are used in the OpenSSL toolkit, to run the executable, the path of the OpenSSL libraries must be added to the `LD_LIBRARY_PATH` environment variable. For more information see [5].

| NOTE | If you are using the test port on Solaris, you have to set the `PLATFORM` macro to the proper value. It shall be `SOLARIS` in case of Solaris 6 (SunOS 5.6) and `SOLARIS8` in case of Solaris 8 (SunOS 5.8). |
|---|---|

# Configuration

The executable test program behavior is determined via the run-time configuration file. This is a simple text file, which contains various sections (for example, `[TESTPORT_PARAMETERS]`) after each other. The usual suffix of configuration files is *.cfg*. For further information on the configuration file see [3].

## LDAP Test Port Parameters in the Test Port Configuration File

The test port uses abstract socket, therefore the abstract socket's parameters also apply when using the LDAP test port. For the parameters of the abstract socket see [7].

Additional test port parameters:

`decode_incoming_message ("yes", "no")`

If this parameter is used with the value `"yes"`, the port will not decode the incoming messages, and sends octetstring messages instead of the `ASP_LDAP_msg` ASP. In case of big incoming messages, this can improve the performance of the Test Port.

This parameter is optional and the default value is `"no"`.

# LDIF support

## External Function for LDIF Import

```
external function f_ImportLDIF(in charstring pl_file_name, boolean pl_resolve_env)
return LDIFData;
```

Importing data from LDIF [8] files into TTCN is possible with the external function `f_ImportLDIF`.

The function has two parameters:

- The name of the file to be opened
- Whether the references to environmental variables should be resolved or passed into TITAN-3 as are.

`f_ImportLDIF` returns a valid structure or prints warnings.

### Error Handling

In case of errors, warnings are written in the RTE log file, and the version in the LDIFData structure is set to `-1`. If no error occurred the version number should be `1`, or omitted if not found in the file.

If an error occurred, the returned LDIFData structure will still contain data to help finding the error.

The data that could be read will be in the structure.

Data that couldn't be read will be in the structure in one of the following ways:

- If the data is optional, it will be omitted.
- If the data is a character string, it's value will be `"ERROR"`.
- If the error occurred inside the element of a list, the element will be filled with values mentioned.
- It the error occurred in a list, the list will have no elements.

## Population, Depopulation

`f_PopulateLDAPServer` and `f_DepopulateLDAPServer` issue LDAP queries based on input gathered with LDIF import external function.

Both functions take the port to communicate on as a parameter. The connection must exist for the functions to work.

Both functions assume that the parameter LDIFData structure is filled in with valid data, therefore they don't check its validity.

### Populate

```
function f_PopulateLDAPServer(LDIFData pl_info, LDAPasp_PT pl_LDAP, EntryConversion
pl_conversion, boolean pl_continue, integer pl_clientID) return boolean;
```

`f_PopulateLDAPServer` performs LDAP operations on each entry of its LDIFData parameter:

- LDAP Add for directory entries and for changerecords with changetype "add".
- LDAP Delete for changerecords with changetype "delete".
- LDAP Modify for changerecords with changetype "modify".
- LDAP ModifyDN for changerecords with changetype "moddn" or "modrdn".

Depending on the value of `pl_conversion` parameter, `f_PopulateLDAPServer` can perform LDAP Modify with "add" (`pl_conversion=Entry2ModifyAdd`) or "replace" (`pl_conversion=Entry2ModifyReplace`) operation instead of the normal LDAP "Add" (`pl_conversion=NoConversion`) for each directory entry.

If the used LDIF data contains directory entries then `f_PopulateLDAPServer` will also merge all

attributes with the same type into a single attribute. The resulting merged attribute will replace the first occurrence of the attribute.

The `pl_continue` parameter can be used to determine whether the function shall continue (`pl_continue=true`) or immediately return (`pl_continue=false`) on error.

The `pl_clientID` parameter identifies the client connection. In case of one connection, this parameter can be set to `-1`.

`f_PopulateLDAPServer` returns true if all LDAP operations concluded successfully. It returns `false` when some LDAP operation failed.

### Depopulate

```
function f_DepopulateLDAPServer(LDIFData pl_info, LDAPasp_PT pl_LDAP, boolean
pl_continue) return boolean;
```

`f_DepopulateLDAPServer` performs LDAP Delete for each entry found in the LDIFData structure, and does nothing for changerecords. Depopulation happens in the opposite order or appearance of directory entries inside the LDIF input.

The `pl_continue` parameter can be used to determine whether the function shall continue (`pl_continue=true`) or immediately return (`pl_continue=false`) on error during LDAP Delete operations.

`f_DepopulateLDAPServer` returns true if all LDAP Delete operations concluded successfully. It returns `false` when some LDAP Delete operation failed.

# Tips and Tricks

# Deviation from the Standard LDAP ASN.1 Type Definition Module

Please read Section 3.4.1 of [4] carefully.

# `f_ImportLDIF` Tips and Tricks

Tips, concerning the parse of LDIF files.

Because of empty or comment lines, the warning message might not tell the exact line number where the error was found. If, for example, an erroneous line is followed by comments, then it might happen, that the last comment line number will be reported as the erroneous line.

There are some cases where a seemingly incorrect warning message is generated. This happens when the reason for error can’t be identified exactly. In these cases try to check for error in a bigger

context. For example, if the warning reports a bad attribute, but all the attributes seem to be correct in the neighborhood of reported line number, you should check if you are using simple content records and change records in the same file.

## Regenerate *lex.ldif.cc* and *ldif.tab.cc/hh*

These files are delivered in the product but it is possible to regenerate them from *ldif.lex* and *ldif.y*.

The scheme of generatation:

*ldif.lex - lex.ldif.cc*

*ldif.yldif.tab.cc/hh*

The commands:

```
flex -Bs –Pldif_ -olex.ldif.cc ldif.lex

bison –d -p ldif_ -o ldif.tab.cc ldif.y
```

# Error Messages

Since the LDAPasp_RFC4511 Test Port uses the Abstract Socket, it can produce the same error messages. For this messages see [7].

`Parameter value <value> not recognized for parameter <name>`

The specified `<value>` in the runtime configuration file is not recognized for the parameter `<name>`.

`inet_ntop() function call failed`

An error occurred while trying to determine the address of the connected client.

`Unknown state while parsing TLV.`

Internal error, occurs if the `decode_incoming_message` Test Port parameter is set to the value `"yes"`.

# Warning Messages

Since the LDAPasp_RFC4511 Test Port uses the Abstract Socket, it can produce the same warning messages. For this messages see [7].

`Unsupported Test Port parameter: <name>`

The test port parameter `<name>` in the runtime configuration file is not supported for this test port.

`get_TVL_length: there was no complete TLV in the buffer from the reading position.`

Occurs only if the `decode_incoming_message` test port parameter is set to the value `"yes"`. It means that there is not a complete TLV in the incoming buffer to send to the test suite. This warning message never occurs.

# Warning Messages Produced by f_ImportLDIF

**`The file <name> could not be opened`**

The named file was not found, or could not be opened for reading.

**`Wrong modification operation name at line <number>`**

The operation name must be `"add"`, `"delete"` or `"replace"`

**`Hyphen expected at line <value>`**

Every modification operation inside a change record must end with a "-".

**`Line number <value> contains more than one data.`**

The line contains too much data, maybe two structures are in the same line.

**`Wrong newsuperior value at line <value>`**

The value of the `newsuperior` attribute is erroneous.

**`The keyword <string> was expected at line <value>.`**

Couldn't find an expected keyword.

**`Wrong deleteoldrdn value at line <value>`**

The value of the `deleteoldrdn` attribute must be `0` or `1`.

**`Wrong newrdn value at line <value>`**

The value of the `newrdn` attribute has errors.

**`Wrong moddn type at line <value>`**

The moddn type must be `"moddn"`, or `"modrdn"`.

**`deleteoldrdn missing from the change moddn structure ending at line <value>`**

The change moddn structures must have a `deleteoldrdn` keyword – value pair inside.

**`Add must be followed by at least 1 attribute:value pair at line <value>`**

The change add structure must have values.

**`Wrong option syntax at line <value>`**

The options of an attribute type are given with wrong syntax.

`Attribute error at line <value>`

The attribute has errors, or is not present.

`<variable> could not be resolved`

The environmental variable could not be resolved. It might not exist or it was mistyped.

`Value error at line <value>`

The value in this attribute – value pair has some errors.

`Attribute must be separated from value with a colon at line <value>.`

The attribute must be separated from the value with a ":".

`ldap_oid error at line <value>`

The `LDAP_OID` has a syntax error.

`Wrong format for the dn at line <value>`

The string following the dn keyword has a syntax error, or is not separated from the dn keyword with a ":".

`Version error at line <value>`

The version string or the version number has a wrong format.

`Content and change records can not be mixed in one file`

There are content and change records in the same file, which is not allowed by the standard.

`The <value> modified attribute's type is different from the one described in the <value> changerecord's <value> modify record (<string>) (<string>)`

The attribute in attribute – value pairs inside change modify records must be the same as the described attribute to be modified.

# Examples

## Configuration File

```
[TESTPORT_PARAMETERS]
// CLIENT settings
*.LDAP_PCO.remote_address := "127.0.0.1"
*.LDAP_PCO.remote_port := "5019"
*.LDAP_PCO.socket_debugging := "YES"
*.LDAP_PCO.ssl_certificate_chain_file := "certificates/CAcert.pem"
*.LDAP_PCO.ssl_private_key_password := "abcd"
*.LDAP_PCO.ssl_private_key_file := "certificates/CAkey.pem"
*.LDAP_PCO.ssl_trustedCAlist_file := "certificates/CAcert.pem"
*.LDAP_PCO.ssl_use_ssl := "yes"
*.LDAP_PCO.use_non_blocking_socket := “yes”

// SERVER settings
*.LDAP_PCO_server.server_mode:="yes"
*.LDAP_PCO_server.local_port:="5019"
*.LDAP_PCO_server.socket_debugging := "YES"
*.LDAP_PCO_server.ssl_certificate_chain_file := "certificates/CAcert.pem"
*.LDAP_PCO_server.ssl_private_key_password := "abcd"
*.LDAP_PCO_server.ssl_private_key_file := "certificates/CAkey.pem"
*.LDAP_PCO_server.ssl_trustedCAlist_file := "certificates/CAcert.pem"
*.LDAP_PCO_server.ssl_use_ssl:="yes"
```

# Makefile

In this section the most important parameters are listed in the *Makefile*. The following gives some detail about them:

`PLATFORM =`

Specifies which platform you are using. If you are using the test port on Solaris, you have to set the `PLATFORM` macro to the proper value. It shall be `SOLARIS` in case of Solaris 6 (SunOS 5.6) and `SOLARIS8` in case of Solaris 8 (SunOS 5.8). In case you are using the test port on other platform, please refer to [2].

`OPENSSL_DIR =`

Specifies the OpenSSL installation directory. It has to contain the *lib/libssl.a* file and the include/ directory.

`CPPFLAGS = -D$(PLATFORM) -DAS_USE_SSL -I$(TTCN3_DIR)/include -I$(OPENSSL_DIR)/include`

This line includes the OpenSSL header files and enables SSL code. It shall be used if SSL is used.

If no SSL is used, the generated *Makefile* by TITAN is suitable.

`LDFLAGS = -lssl`

This line specifies the OpenSSL runtime library. It shall be used if SSL is used.

`TTCN3_MODULES =`

The list of TTCN-3 modules needed.

```
USER_SOURCES = LDAPasp_PT.cc Abstract_Socket.cc ldif.tab.cc lex.ldif.cc
```

```
USER_HEADERS = $(USER_SOURCES:.cc=.hh)
```

The list of other external C++ source and header files.

# Example Use of ImportLDIF Function

```
module LDAPtest {

import from LDAPasp_PortType all;
import from LDAPasp_Types all;
import from Lightweight_Directory_Access_Protocol_V3 language "ASN.1:1997" all;
import from LDIF all;
type component LDAPcomp_CT {
  port LDAPasp_PT LDAP;
}

testcase tc1() runs on LDAPcomp {

  map(system:LDAP, self:LDAP);
  timer T2 := 100.0;
  T2.start;

  var LDIFData information;
  information := f_ImportLDIF("test1.ldif",false);
  f_PopulateLDAPServer(information,false,LDAP);


  setverdict(pass);
  timer T3 := 3.0;
  T3.start;
  T3.timeout;

  unmap(system:LDAP, self:LDAP);
}
}
```

# Terminology

**OpenSSL:**
The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and open source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. For more information on the OpenSSL project see [5].

# Abbreviations

**ASP**
: Abstract Service Primitive

**ES**
: ETSI Standard

**ETSI**
: European Telecommunications Standards Institute

**IUT**
: Implementation Under Test

**LDAP**
: Lightweight Directory Access Protocol

**LDIF**
: LDAP Data Interchange Format

**RTE**
: RunTime Environment

**SSL**
: Secure Socket Layer

**SUT**
: System Under Test

**TCP**
: Transmission Control Protocol

**TTCN-3**
: Testing and Test Control Notation version 3

# References

[1] ETSI ES 201 873-1 v3.1.1 (06/2005)
The Testing and Test Control Notation version 3. Part 1: Core Language

[2] Installation Guide for TITAN TTCN-3 Test Executor

[3] Programmer’s Technical Reference for the TITAN TTCN-3 Test Executor

[4] LDAPasp_RFC4511 Test Port for TTCN-3 Toolset with TITAN, Function Specification

[5] OpenSSL toolkit

http://www.openssl.org

[6] RFC 4511
Lightweight Directory Access Protocol v3

[7] Abstract Socket Test Port for TTCN-3 Toolset with TITAN, User Guide

[8] RFC 2849
The LDAP Data Interchange Format (LDIF) – Technical Specification