

Source: Franck Régnier / KEREVAL
<mailto:franck.regnier@kereval.com>

Title: DHCPv6 test suite developed by GO4IT

Document for:

Decision	
Discussion	
Information	X

Introduction of GO4IT

The GO4IT project will provide a generic low cost test environment and associated test services, together with free of charge executable test suites, targeting IPv6 focused protocol testing. The GO4IT project will permit to capitalize on large European investments made in infrastructures by giving the tools to all user communities to test IPv6 protocol based solutions thus increasing their trust and confidence in new generation infrastructures.

GO4IT test platform consists of two sets of tools for IPv6 testing:

- **Package1** tools: freely downloadable and ready to use test suites for IPv6 protocols, based on:
 1. ETSI test suites for IPv6: TTCN-3 source code (Abstract Test System) and documentation publicly available on ETSI Internet Protocol Testing web site for download.
 2. Free test tools, based on Java/Testing Tech technology, dedicated to ETSI Test suite execution, and available for download on GO4IT web site, with associated documentation and online service.
- **Package2** tools: TTCN-3 test environment (IPv6 test repository, edition environment, compiling tools, and execution environment), developed by Go4IT project partners, open to collaborative work, and available under open source licence on Go4IT web site.

Package2 roadmap is structured into 3 steps:

1. *Go4IT Project Test Platform (short term, project Duration)*: This is the platform concept upon which the Go4IT project partners will develop Package2 deliverables. It supports relatively simple IPv6 protocol ATS using only a subset of TTCN-3 functionality. The checked items in the corresponding column of the table below indicate this subset. It is derived, in part, from the INRIA RIPng ATS. However, additional considerations have also been included.
2. *Optimum Go4IT Test Platform*: It supports all IPv6 protocol ATS. Again, it is a subset of TTCN-3 functions but has more functionality than the Project Test Platform. This concept is derived in part from the ETSI STF 276 IPv6 Protocol ATS for Core, IPsec, Mobility, and Transition services. Again, additional considerations are included.
3. *Long Term Test Platform (long term view)*: This is our view of how the Go4IT could evolve over time with in a Source Forge-like development environment. It includes additional functionality that is not necessary for IPv6 protocol testing.

Abbreviations and Acronyms

NUT	Node Under Test
DHCP	Dynamic Host Configuration Protocol
DUID	DHCP Unique Identifier
ATS	Abstract Test Suite
CD	CoDec (Coding/Decoding)
IUT	Implementation Under Test (part of the system being tested)
PA	Platform Adapter
SA	System (SUT) Adapter
SUT	System Under Test
TC	Test Case
TCI	TTCN-3 Control Interface
TE	TTCN-3 Test Executable
TRI	TTCN-3 Runtime Interface
TTCN-3	Testing and Test Control Notation version 3

References

- Request for Comments 3315 - Dynamic Host Configuration Protocol for IPv6 (DHCPv6), July, 2003.
- Request for Comments 3646 - DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6), December, 2003
- Request for Comments 3736 - Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6, April, 2004.
- IPv6 READY Phase II Test Specification DHCPv6 Revision 1.0.0

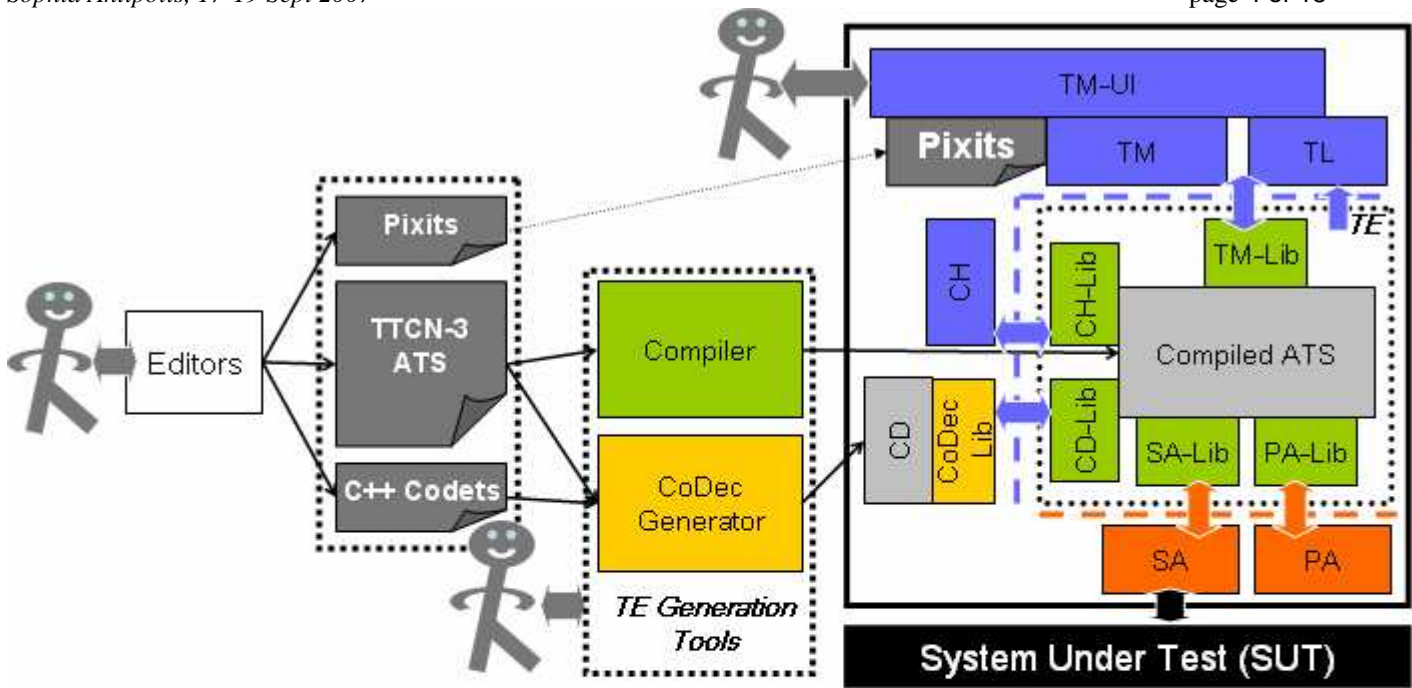
System Overview











The objectives and constraints of this Package2 are:

- Provide tool independent, extendable, "Go4IT ATS" (minimum tests suites, types, templates, functions) as a validation tool for the CoDec and adapter (PA, SA) implementations. This ATS will be TTCN-3 compliant.
- Provide, as source code, all the TTCN-3 modules that are bonded to the ATS (CoDec, SA, PA), for this "Go4IT ATS".
- Provide, as source code, the modules that are independent from the ATS (generic) and are needed for the test execution of any TTCN-3 compliant ATS.
- Provide a TTCN-3 compiler (adapted to the Go4IT ATS specific limitations) and associated runtime libraries.
- Provide a validation module (source code) for each Go4IT software module. This will allow independent validation of each module for both the user need verification (the specified functions are implemented) and the TTCN-3 standard compliance validation (TRI, TCI, operational semantics, core language).
- Define the set of reference documents (TTCN-3 version 3): provide and maintain the list of supported mechanisms in each of the developed module.
- Provide open documentation and free access to the platform (source forge like)

Go4IT architecture is compliant to TTCN-3 standard definitions of the TRI and TCI interfaces. The Go4IT architecture is conforming to TTCN-3 standard definitions. By respecting such a basic modular architecture and the detailed specifications for the internal interfaces between modules (TRI and TCI), Go4IT will provide an open platform for TTCN-3 development and test execution.

GO4IT test platform is depicted on the following figure.



-  GO4IT User
-  GO4IT ATS, including: TTCN-3 ATS, C++ Codets (TTCN-3 types extensions), Pixit
-  TRI API
-  TCI API
-  Adapters (developed by GO4IT partners, depend on the ATS)
-  Generated code (from the ATS)
-  Generic Modules (independant from the ATS)
-  Compiler and Runtime libraries (independant from the ATS)
-  CoDec Generator and Runtime librairie (independant from the ATS)
-  Editor (third party tool)

DHCPv6 Overview

The Dynamic Host Configuration Protocol for IPv6 (DHCP) enables DHCP servers to exchange configuration parameters such as IPv6 network addresses to IPv6 nodes. It offers the capability of automatic allocation of reusable network addresses and additional configuration flexibility. DHCP is a client/server protocol that provides managed configuration of devices.

The targeted protocol is specified through the following documents:

- RFC 3315 Dynamic Host Configuration Protocol for IPv6 (DHCPv6), July, 2003.
- RFC 3646 DNS Configuration options for DHCPv6, December, 2003
- RFC 3736 Stateless Dynamic Host Configuration Protocol Service for IPv6, April, 2004.

Three devices are defined: Client, Server and Relay-Agent

- DHCP client (or client): A node that initiates requests on a link to obtain configuration parameters from one or more DHCP servers.
- DHCP relay agent (or relay agent): A node that acts as an intermediary to deliver DHCP messages between clients and servers, and is on the same link as the client.
- DHCP server (or server): A node that responds to requests from clients, and may or may not be on the same link as the client(s).

Strategic choices

The development of the hot topic ATS will follow the test specification published by the IPv6 Ready Committee and focus on the Client profile. Additional test cases will be defined to test DHCPv6 features that are not covered by the logo.

The current version is 1.0.0 and was published in April 2007.

There are several reasons for these choices

- The IPv6 Ready Logo is widely recognized in the area of IPv6 testing. Using its test specification will contribute to promote TTCN-3 testing in the Internet Community.
- The test specification defines three profiles of devices: Client, Server and Relay-Agent. It is expected that the number and diversity of Client devices in the market will be higher (eg. embedded devices).
- Using an existing test specification will allow to cover fully at least one device profile and release an executable test suite within the duration of the project.

Here is a table giving in detail number of test cases corresponding to each group of tests as defined by the IPv6 Ready Logo Program.

Number of Testcases	Total
Section 1 - RFC3315 Client	
Group 1	20
Group 2	33
Group 3	36
Section 4 - RFC3646 Client	18
Section 7 - RFC3736 Client	26
Section 2 - RFC3315 Server	
Group 1	24
Group 2	12
Group 3	46
Section 5 - RFC3646 Server	12
Section 8 - RFC3736 Server	33
Section 3 - RFC3315 Relay Agent	42
Section 6 - RFC3646 Relay Agent	36
Section 9 - RFC3736 Relay Agent	22
Total	360

According to the work plan the development covers only the Client part and provides:

- Client Abstract Test Suite in TTCN-3
 - sections 1, 4 and 7
- Executable test suite using the GO4IT Package2 environment
- Validation of the test suite to ensure the quality of the final result
 - code Review and execution under a implementation
- Additional test cases that are not part of the IPv6 Ready Logo

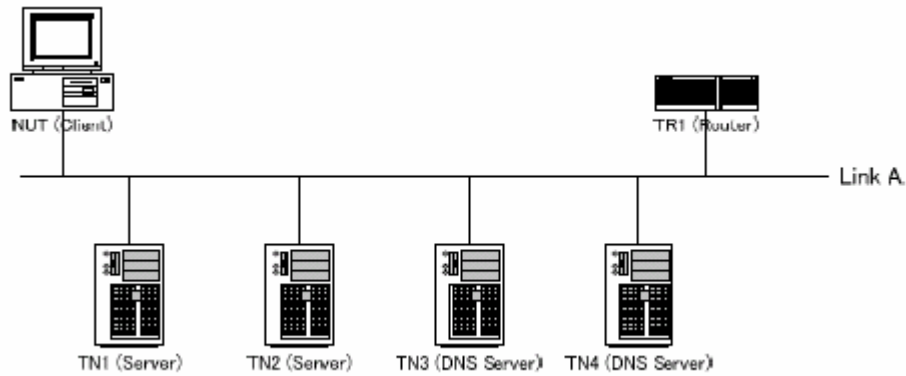
The scope of the test includes major functionality groups such as client behaviour in client-initiated configuration exchange, client behaviour in server-initiated configuration exchange, client behaviour in server solicitation, and message validation by client.

The tests verify the process for receiving a list of available DNS recursive name servers and a domain search list from a server in parallel with Address Assignment and from a server in Stateless Dynamic Host Configuration Protocol for IPv6.

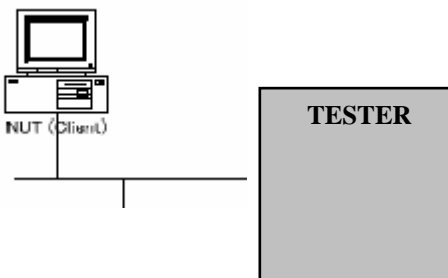
These tests are designed to verify the readiness of a DHCPv6 client implementation vis-à-vis the base specifications and the DNS Configuration options of the Dynamic Host Configuration Protocol for IPv6.

Network Topology

Common Topology (Client test cases)



Topology used: 1 NUT Client , 1 Link, 1 Tester to emulate all the nodes on the network



Sample Test Result

No.	Time	Type	Source	Destination	Protocol	Info
1	0.000000				Testcase	DHCPv6-1.2.3.4 Renew message format
2	2.425027	<->	NUT	All DHCP Relay Agents	DHCPv6	Solicit
3	2.423807	-->	TN1	All	DHCPv6	Advertise
4	3.443823	<->	NUT	All DHCP Relay Agents	DHCPv6	Request
5	3.450441	-->	TN1	All	DHCPv6	Reply
6	3.461549				Verdict	Step 1: Pass (Finished Client Common Test)
7	3.486109	<->	NUT	?	ICMPv6	Multicast Listener Report Message v2
8	3.598010	<->	NUT	?	ICMPv6	Neighbor solicitation
9	3.598490				Verdict	Step 2: Pass
10	11.126325	<->	NUT	?	ICMPv6	Multicast Listener Report Message v2
11	53.466696	<->	NUT	All DHCP Relay Agents	DHCPv6	Renew
12	53.467360				Verdict	Step 3: Pass
13	53.498518				Verdict	Step 3: Pass (The Renew message contains
14	53.499608				Verdict	Step 3: Pass (The Renew message contains
15	53.502954				Verdict	Step 3: Pass (Testcase Finished)

Abstract Test Suite Example

Test DHCP_CONF.1.2.1: Transmission of Solicit Messages

Purpose: To verify a client device transmits properly formatted Solicit messages and properly follows the retransmission algorithm for Solicit messages.

References:

- [DHCP 3315] – Sections 5.5, 14, 15.1, 16, 17.1, 17.1.1 and 17.1.2

Test Setup: Connect the network according to the Common Topology. DHCPv6 is enabled on the client device before each part. DHCPv6 on the client device is disabled after each part.

Procedure:

Part A: Solicit message format

1. Enable DHCPv6 on the NUT.
2. Observe the first Solicit message transmitted on Link A.

Part B: Reliability of DHCPv6 Retransmission

3. Enable DHCPv6 on the NUT.
4. Observe the first Solicit message transmitted on Link A.
5. Wait for second Solicit message.
6. Observe the second Solicit message transmitted on Link A.

Part C: Retransmission of Solicit Message

7. Enable DHCPv6 on the NUT.
8. Observe the time the first Solicit message was transmitted on Link A.
9. Wait for second Solicit message.
10. Observe the time the second Solicit message was transmitted on Link A.

Part D: Maximum Retransmission Time of Solicit Message

11. Enable DHCPv6 on the NUT.
12. Continue to capture Solicit messages until $RT_{prev} = MRT + MRT * RAND$ ($108 \leq RT_{prev} \leq 132$).
13. Observe the messages transmitted on Link A.

Observable Results:

• *Part A*

Step 2: The NUT transmits a properly formatted Solicit message containing the following elements:

- Src Address is a link-local for that interface
- The msg-type field was set to the value of 1 (Solicit)
- A header containing a Transaction ID

• *Part B*

Step 4: The NUT transmits a properly formatted Solicit message containing the following elements:

- Src Address is a link-local for that interface
- The msg-type field was set to the value of 1 (Solicit)
- A header containing a Transaction ID
- A Client Identifier Option (containing a DUID)

Step 6: The NUT transmits a properly formatted Solicit message with the same values as in Step 4. The transaction ID is the same for all retransmitted messages.

• *Part C*

Step 8: The NUT transmits a properly formatted Solicit message.

Step 10: The NUT transmits a properly formatted Solicit message according to the Second message in the chart below.

Solicit Message	Minimum Delay	Maximum Delay
First message	-	-
Second message	RT (greater than 1.0)= IRT + RAND*IRT Where IRT=1, RAND>0	RT(1.1) = IRT + RAND*IRT Where IRT=1, RAND= +.1

• *Part D*

Step 13: The NUT should properly transmit Solicit messages according to the chart below. The transaction ID is the same for all retransmitted messages.

Solicit Message	Minimum Delay	Maximum Delay
First message	-	-
Second message	RT (greater than 1.0 sec)= IRT + RAND*IRT Where IRT=1, RAND>0	RT(1.1 sec) = IRT + RAND*IRT Where IRT=1, RAND= +.1
X message	108 seconds =.9*MRT where MRT=120	132 seconds =1.1*MRT where MRT=120
X+1 message	108 seconds =.9*MRT where MRT=120	132 seconds =1.1*MRT where MRT=120

ATS File: DHCP_CONF_1_2.TTCN**template** FrameType DHCP_CONF_1_2_1_DHCPv6Solicit_CheckMessageFormat_rtp

(template DHCPv6OptionListType options, TL_Type src)

modifies DHCPv6Solicit_rtp :=

```
{
    [1] := { ipv6 := ipv6_rtp (src.LLocal_addr, ?, ?)},
    [3] := {dhcpv6:= dhcpv6_Solicit_tp (?, options)}
}
```

testcase DHCP_CONF_1_2_1_A ()

runs on TestComponent

system TestSystem

```
{
    var FrameType frame := {} ;

    TestBegin ("DHCP_CONF.1.2.1.A Solicit message format");
    activate (Defaults_Altstep());
    activate (NS_Altstep());
    activate (NoDHCPv6Messages_Altstep());

    Step := "1";
    action ("Disable DHCPv6 on the NUT, then press Enter and enable DHCPv6 on the NUT");
    link1.clear;

    Step := "2";
    Receive (1, frame, 60.0,DHCP_CONF_1_2_1_DHCPv6Solicit_CheckMessageFormat_rtp (*, NUT1));

    TestEnd();
}
```

template FrameType DHCP_CONF_1_2_1_B_DHCPv6Solicit_CheckMessageFormat_SameTransactionId_rtp

(template DHCPv6OptionListType options, TL_Type src, FrameType frame)

modifies DHCP_CONF_1_2_1_DHCPv6Solicit_CheckMessageFormat_rtp :=

```
{
    [3] := {dhcpv6:= dhcpv6_Solicit_tp (frame[3].dhcpv6.TransactionId, options)}
}
```

testcase DHCP_CONF_1_2_1_B ()

runs on TestComponent

system TestSystem

```
{
    var FrameType frame := {};
    var FrameType frame2 := {};
    var DHCPv6OptionType client_id;

    TestBegin ("DHCP_CONF.1.2.1.B Reliability of DHCPv6 Retransmission");
    activate (Defaults_Altstep());
    activate (NS_Altstep());
    activate (NoDHCPv6Messages_Altstep());

    Step := "3";
    action ("Disable DHCPv6 on the NUT, then press Enter and enable DHCPv6 on the NUT");
    link1.clear;

    Step := "4";
    Receive_noverdict (1, frame, 60.0,DHCP_CONF_1_2_1_DHCPv6Solicit_CheckMessageFormat_rtp (?,
    NUT1));

    If (DHCPv6FindUniqueOption (frame, DHCPv6OptionClientId, client_id)) {
        LogVerdict (pass, "The Solicit message contains a Client Identifier Option");
    } else {
        LogVerdict (fail, "The Solicit message doesn't contain any Client Identifier Option");
    }
}
```

```
    Step := "5";

    Receive (1, frame, 20.0,
DHCP_CONF_1_2_1_B_DHCPv6Solicit_CheckMessageFormat_SameTransactionId_rtp ({client_id,*}, NUT1,
frame));

    TestEnd();
}

```

testcase DHCP_CONF_1_2_1_C ()

runs on TestComponent

system TestSystem

```
{
    var FrameType frame := {};
    var FrameType frame2 := {};
    var float timer_temp := 0.0;
    var float timer_temp2 := 0.0;
    var float temp := 0.0;

    TestBegin ("DHCP_CONF.1.2.1.C Retransmission of Solicit Message");
    activate (Defaults_Altstep());
    activate (NS_Altstep());
    activate (NoDHCPv6Messages_Altstep());

    Step := "7";

    action ("Disable DHCPv6 on the NUT, then press Enter and enable DHCPv6 on the NUT");
    link1.clear;

    timer time := 10.0;
    time.start;

    Step := "8";

    Receive (1, frame, 2.0,DHCPv6Solicit_rtp (?));

    timer_temp := time.read;

    Step := "9";
}

```

```
Receive_noverdict (1, frame2, 2.0,DHCPv6Solicit_rtp (?));
```

```
timer_temp2 := time.read;
```

```
time.stop;
```

```
temp := timer_temp2 - timer_temp;
```

```
Step := "10";
```

```
if((temp > 1.0) and (temp < 1.1)) { //minimum and maximum delay
```

```
    LogVerdict(pass,"The NUT transmits a properly formatted Solicit Message in the expected time range");
```

```
  } else {
```

```
    LogVerdict(fail,"The NUT doesn't transmit any properly formatted Solicit Message in the expected time range");
```

```
  }
```

```
  TestEnd();
```

```
}
```

```
testcase DHCP_CONF_1_2_1_D ()
```

```
runs on TestComponent
```

```
system TestSystem
```

```
{
```

```
  var FrameType frame := {} ;
```

```
  var FrameType frame2 := {} ;
```

```
  var DHCPv6OptionType client_id;
```

```
  var float prev_tempo := 0.0;
```

```
  var float tempo := 0.0;
```

```
  var float tempo2 := 0.0;
```

```
  TestBegin ("DHCP_CONF.1.2.1.D Maximum Retransmission Time of Solicit Message");
```

```
  activate (Defaults_Altstep());
```

```
  activate (NS_Altstep());
```

```
  activate (NoDHCPv6Messages_Altstep());
```

```
Step := "11";
```

```
action ("Disable DHCPv6 on the NUT, then press Enter and enable DHCPv6 on the NUT");
```

```
link1.clear;
```

```
Step := "12";
```

```
timer time := 400.0;
```

```
Receive_noverdict(1, frame, 60.0, DHCPv6Solicit_rtp (?));
```

```
time.start;
```

```
DHCPv6FindUniqueOption (frame, DHCPv6OptionClientId, client_id);
```

```
Receive_noverdict(1, frame2, 2.0,  
DHCP_CONF_1_2_1_B_DHCPv6Solicit_CheckMessageFormat_SameTransactionId_rtp ({client_id,*}, NUT1,  
frame));
```

```
prev_tempo := time.read;
```

```
if((prev_tempo < 1.0) or (prev_tempo > 1.1)) { //minimum and maximum delay
```

```
    LogVerdict(fail, "Second message was not sent within the expected range");
```

```
}
```

```
while (time.running) {
```

```
Receive_noverdict(1, frame2, 132.0,  
DHCP_CONF_1_2_1_B_DHCPv6Solicit_CheckMessageFormat_SameTransactionId_rtp ({client_id,*}, NUT1,  
frame));
```

```
tempo := time.read;
```

```
if (((tempo - prev_tempo) >= 108.0) and ((tempo - prev_tempo) <= 132.0)) {
```

```
    goto finished
```

```
}
```

```
prev_tempo := tempo;
```

```
}
```

```
label finished;
```

```
Step := "13";
```

`tempo2 := time.read;``Receive_noverdict(1, frame2, 132.0,
DHCP_CONF_1_2_1_B_DHCPv6Solicit_CheckMessageFormat_SameTransactionId_rtp ({client_id,*}, NUT1,
frame));``tempo := time.read;``if(((tempo - tempo2) >= 108.0) and ((tempo - tempo2) <= 132.0)){``LogVerdict(pass,"");``} else {``LogVerdict(fail,"");``}``TestEnd();``}`