Chapter

8 *Tutorial: The TTCN Link*

This is a tutorial for TTCN Link. It is intended to give a guided tour through some of the facilities provided.

To be able to follow this tutorial, you need some experience of both SDL and TTCN as well as the SDL suite and TTCN suite.

Purpose of This Tutorial

The purpose of this tutorial is to make you familiar with the TTCN Link function and understand its usability. You are supposed to read this tutorial sequentially and practice the exercises.

> **Note: Platform differences**
>
> This tutorial is possible to run on both the UNIX and Windows platform, and is described in a way common to both platforms. In case there are differences between the platforms, this is indicated by texts like “on UNIX”, “Windows only”, etc.
>
> When such platform indicators are found, please pay attention only to the instructions for the platform you are running on.

More about TTCN Link

TTCN Link is a CATG (Computer Aided Test Generation) tool that enables efficient development of TTCN test suites based on SDL specifications. By using TTCN Link, you can automatically generate the declarations from the SDL specification and interactively build test cases. This ensures consistency between the test suite and the specification. TTCN Link also provides you with an integrated SDL/TTCN environment with access to the specification directly from the test case.

Using TTCN Link

The inputs needed to use TTCN Link for test case development, are an SDL specification and a test suite structure with test purposes. TTCN Link supports test suite development in three different ways:

- Keeping the test suite consistent with the SDL specification.
- Supporting the test suite developer with information about the SDL specification, making the actual TTCN coding easier.
- Providing access to the SDL specification directly from the test case.

Three distinct phases can be distinguished when TTCN Link is used:

- Preparation of the SDL specification to be used in the test case development.
- Generation of the TTCN declarations.
- Development of the TTCN test cases and constraints.

Taking a Look at the SDL System

In this tutorial, you will use the example SDL system called `inres` (short for Initiator-Responder). This system is an example of a simplified communication protocol intended to give a secure transfer of information over an unsafe communication medium. It provides a one-way, connection-oriented communication service that uses sequence numbers and retransmission to ensure that messages sent to the initiator side, are correctly received at the responder side.

Start this tutorial by taking a look at the `inres` SDL system:

1. Copy the `inres` directory from `$telelogic/sdt/examples/` **(on UNIX)** or `\Telelogic\SDL_TTCN_Suite4.5\sdt\examples\` **(in Windows)** to your working directory and make sure that you have write permission.

2. Start Telelogic Tau.

3. Open the example SDL system called `inres.sdt` located in the `inres` directory.

4. Select *SDL Overview* in the *Generate* menu in the Organizer, to generate an overview of the SDL system.

   The *Generate SDL Overview* dialog is opened. You do not have to change any settings in this dialog.

5. Click the *Generate* button in the *Generate SDL Overview* dialog.

   The SDL Editor is opened and displays the overview.

   The block *Station_Ini* in *System inres* is the protocol that you are going to create a test for. However, due to the test architecture that is to be used, the tester can not access the lower service access point of the protocol. Since TTCN Link assumes that the channels to/from the environment are accessible for the tester (and will correspond to

PCOs in the test suite), the block *Medium*, that models the underlying service provider, is included in the SDL system.

Generating a TTCN Link Executable

To create a test suite based on an SDL system, it is necessary to begin by creating an executable program that will be used during the test suite development. Such programs, which are created by the SDL to C Compilers, are called *Link Executables*. Sometimes they will also be referred to as *state space generators*, since the purpose of these programs is to generate the combined state space of the SDL system and the TTCN test case.

1. Select *Make* in the *Generate* menu in the Organizer, to create a Link Executable for the SDL system.

   – Since you probably have not saved the system since you generated the overview, you will be prompted to do so before the *SDL Make* dialog can be opened.

2. Make sure that the following options are set in the *SDL Make* dialog:

   – *Code generator: Cbasic*
   – *Prefix: Full*
   – *Separation: No*
   – *Capitalization: Lower case*

3. Make also sure that the following options are turned on:

   – *Analyze & generate code*
   – *Makefile*
   – *Compile and link*

4. Select *Generate Makefile*.

5. Select a *TTCN Link* kernel for a suitable compiler in the *Use standard kernel* option.

6. Click the *Full Make* button.

   A TTCN Link executable will now be generated from the SDL specification. You can open the Organizer log to see the result of the make process.

Now the preparations are completed and you can start developing a new test suite.

Creating a Test Suite

What You Will Learn

- To specify the state space generators to use, in two different ways
- To generate the TTCN declarations

Creating a New Test Suite

Start by creating a new test suite:

- Add a new test suite, for example called `inres`, to the *TTCN Test Specification* chapter. In the *Add New* dialog, make sure that the option *Show in editor* is on.

  The TTCN suite is started and the Browser will display the new test suite.

The first step when you have created the new test suite, is to use TTCN Link to generate the declarations. Before you can do this, you have to tell the TTCN suite what state space generator to use.

Specifying the Link Executable

Before you can generate the declarations, you have to tell the TTCN suite which link executable to use. You can do this in two ways:

One way is to associate the test suite with the SDL system:

1. In the Organizer, select the top icon in *System inres*.
2. Select *Associate* from the *Edit* menu.

   The *Associate* dialog is opened.
3. Select the test suite that you have created.
4. Click the *OK* button.

   The association is illustrated by a new icon placed under the test suite icon.

Another way is to explicitly specify the state space generator to use:

1. In the TTCN suite, select *Select Link Executable* from the *SDT Link* menu. (**On UNIX**, it is the menu in the Browser.)

   The *Select Link Executable* dialog is opened.

2. Select `inres_stx.exe`.

   All generators will by default be given the name `<sdl system name>_st<x>.exe` , where `<x>` is depending on which compiler was used when the executable was created.

3. Click *OK*.

> **Note:**
> In case a link executable has been selected both in the Organizer and in the TTCN suite, the one selected in the TTCN suite is the link executable that will be used.

Generating the TTCN Declarations

Since the SDL specification contains all information about the interfaces, this information is available to you when using TTCN Link. When you generate the TTCN declarations, the interface information is extracted from the SDL specification and included in the TTCN test suite as PCO, ASP and data type tables.

To generate the TTCN declarations:

1. In the TTCN suite, select *Generate Declarations* from the *SDT Link* menu. (**On UNIX**, it is the menu in the Browser.)

2. Expand the declarations part of the test suite and have a look at the generated declarations.

   Objects of the following types have been generated:

   – ASN.1 type definitions
   – PCO type declarations
   – PCO declarations
   – ASN.1 ASP type definitions

   The PCO declarations, and the PCO type declarations, are generated from the channels to/from environment in the SDL system.

The ASP definitions are generated from the SDL signals that can be transported on the channels to/from the environment.

The ASN.1 type definitions are generated from the types of the parameters of the above mentioned signals.

Generated declarations will be analyzed automatically.

3. Expand the dynamic part of the test suite.

   A defaults library is generated. It contains the table *OtherwiseFail* with an *otherwise statement* for each PCO, and in addition a timeout statement that match any timeout event.

Creating a Simple Test Case

It is now time to create the first test case. The one you will develop is the test case intended to test the successful connection establishment of the protocol.

How to create constraints and test cases with TTCN Link differs between Windows and UNIX. **On UNIX**, it is possible to create constraints directly from the *Send* dialog. **In Windows**, you have to create the constraints first, before they will be visible in the *Link* dialog.

What You Will Learn

- To synchronize a test case **(UNIX only)**
- To create constraints
- To add send statements
- To add receive statements

Creating a Test Case and Constraints (in Windows)

Before you start creating the test case, you have to create a constraint for the ASP `ICONreq`:

1. Copy the table *ICONreq* from *ASN.1 ASP Type Definitions*.
2. Paste the table in the constraints part, below *ASN.1 ASP Constraint Declarations*.
3. Open this table and change the name to, for example, `ICONreq_1`.
4. Analyze the table.
5. Close the table.

Creating a Test Case and Adding Statements

You are now going to create a test case and add send and receive statements by using the *Link* dialog. When you use the *Link* dialog, the test case is synchronized with the SDL specification and you cannot edit the behavior lines directly in the table.

1. Create a new test case, for example called `test_case_1`.
2. Open the test case in the Table Editor.

3. Select *Link* from the *SDT Link* menu.

   The *Link* dialog is opened. The first thing you will add is a send statement for the ICONreq service primitive.

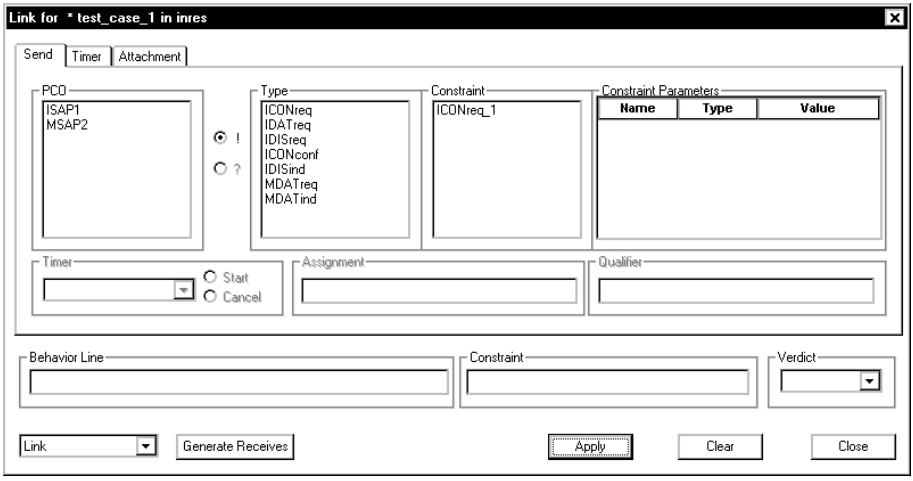

Figure 68: The Link dialog

4. In the dialog, select the following:
   – The PCO ISAP1
   – The type ICONreq
   – The constraint ICONreq_1

   The contents of the *Behavior line* and *Constraint* fields will be updated in accordance with your selection.

5. Click *Apply*.

   The following behaviour line, a send statement, is added to the test case:

Behaviour Description	Constraints Ref
`ISAP1 ! ICONreq`	`ICONreq_1`

6. Click *Generate Receives*.

   This will add a second behavior line, a receive statement, to the test case:

Behaviour Description	Constraints Ref
ISAP1 ! ICONreq	ICONreq_1
MSAP2 ? MDATind	MDATind500

A corresponding constraint will also be created.

If needed, just move the *Link* dialog to see the test case. **Do not close** the dialog.

The table MDATind500 (that was just created) is made visible in the Browser.

7. Analyze the test suite by selecting *Analyze Suite* from the *Build* menu.

Creating Another Constraint

Before you can complete the connection establishment, by adding send and receive statements for the MDATreq ASP, you have to create a new constraint:

1. In the Browser, create a copy of the constraint that was just created, MDATind500.
2. Open the copy of MDATind500 and make the following changes:
   - The name should be MDATreq_1
   - The ASP type should be MDATreq
   - The “id” part of the parameter should be CC instead of CR, which means that the constraint value should be:

```
{ mSDUType1 {id CC,
  num zero,
 data 0}}
```

3. Analyze the table.
4. Close the table.

Adding Statements to Complete the Connection

1. In the *Link* dialog, select *Clear*.

   This will refresh the dialog.

2. Select the following:

   – The PCO MSAP2
   – The type MDATreq
   – The constraint MDATreq_1

3. Click *Apply*.

   This will add a third behaviour line to the test case.

4. Click *Generate Receives*.

   The test case should now contain 4 behavior lines.

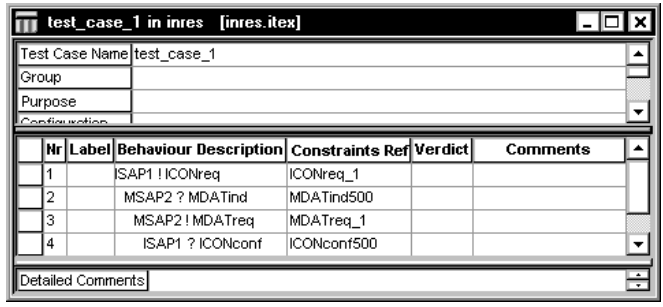

Figure 69: The complete test case

5. Close the *Link* dialog.

Creating a Test Case and Constraints (on UNIX)

1. Create a new test case, for example called `test_case_1`.
2. Open the test case in the Table Editor.

   As you can see, the Table Editor contains the extra menu *SDT Link*.
3. Select *Resynchronize* from the *SDT Link* menu.

   The test case will now be synchronized with the SDL specification. This means that the behaviour lines will be modified from the *SDT Link* menu. However, if you edit a field in the test case manually – except for comment fields – the editor will not be synchronized anymore.

   What happens behind the scene is that the state space generator is started and an initial part of the state space is explored. However, you do not have to think of that now.

Adding Statements

The first thing you will add, is a send statement for the ICONreq service primitive:

1. Select *Send* in the *SDT Link* menu. The *Send* dialog is displayed:

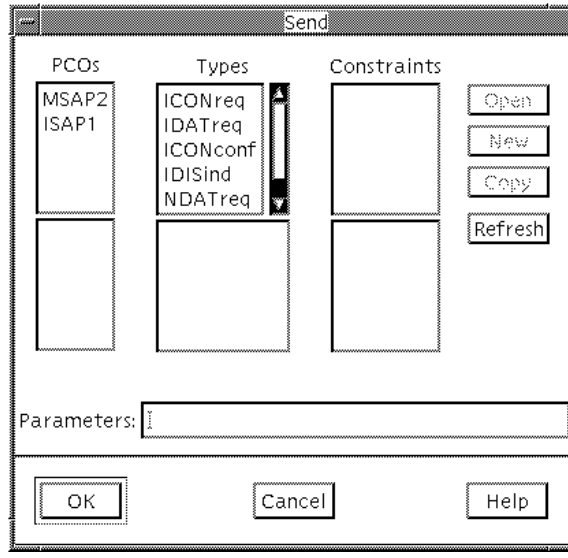

Figure 70: The Send dialog

2. Select *ISAP1* in the *PCOs* list.

   The unselected PCO is moved to the lower part of the list.

3. Select *ICONreq* in the *Types* list.

   The unselected types are moved to the lower part of the list.

4. Click the *New* button to create a constraint for this ASP.

   A Table Editor, displaying the new constraint is opened.

5. Change the name of the constraint to, for example, **`ICONReq_1`**.

6. Analyze the table.

7. Close the table.

8. Click the *Refresh* button in the *Send* dialog to make the new constraint, ICONReq_1, visible in the *Constraints* list.

   Since this is the only constraint of the selected type, it will automatically be selected.

9. Click *OK*.

   The following behaviour line, a send statement, is added to the test case:

Behaviour Description	Constraints Ref
`ISAP1 ! ICONreq`	`ICONreq_1`

10. Select *Receive* from the *SDT Link* menu.

    This adds second behavior line, a receive statement, to the test case:

Behaviour Description	Constraints Ref
`ISAP1 ! ICONreq`	`ICONreq_1`
`MSAP2 ? MDATind`	`MDATind500`

The corresponding constraint is created and analyzed.

Adding Statements to Complete the Connection

To complete the connection establishment, you should now add send and receive statements for the MDATreq ASP:

1. Select row 2 in the *Behaviour Description* column – the field containing `MSAP2 ? MDATind`.
2. Select *Send* from the *SDT Link* menu.

   The *Send* dialog will once again be opened. Now you will create a new constraint by copying an old constraint and modify it.
3. Select *MDATind500* in the *Constraints* list.
4. Click *Copy*.

   The Table Editor will appear, displaying a copy of the MDATind500 constraint.
5. Change the name to `MDATreq_1`.
6. Change the ASP type to `MDATreq`.
7. Change the “id” part of the parameter to `CC` instead of `CR`, which means that the constraint value should be:

```
{ MSDUType1 {id CC,
  num zero,
 data 0}}
```

8. Analyze the table and then close it to get back to the *Send* dialog.
9. Click *Refresh* in the *Send* dialog.
10. Select the new constraint, MDATreq_1.
11. Select the PCO MSAP2.
12. Click the *OK* button.

    The new send statement is inserted in the test case table.
13. Select *Receive* from the *SDT Link* menu once more to complete the connection establishment test case.

Taking a Look at the SDL System (Again)

What You Will Learn

- To display an SDL diagram
- To display an MSC diagram

Displaying an SDL Diagram

It is possible to display the SDL system directly from the test case:

1. Select a behaviour line in the test case.
2. Select *Show SDL* from the *SDT Link* menu.

   The SDL Editor is started, displaying all symbols in the SDL process graphs that where executed in the SDL system as a response to the currently selected row in the test case.

Displaying an MSC Diagram

It is also possible to display an MSC diagram, to illustrate and document the test case:

1. Select a behaviour line in the test case.
2. Select *Show MSC* from the *SDT Link* menu.

   The MSC Editor is started, displaying the execution of the SDL system state up to the currently selected row in the test case. This may be useful as a means to understand how unexpected receive statements are possible.

So Far...

You should now have learned how to create a link executable and how to tell the TTCN suite which link executable to use. You should also know how to generate TTCN declarations from SDL specifications and how to create a test case that is synchronized with SDL specifications. Finally, you should have tried to generate SDL and MSC diagrams from the test case.

If you know how to use the SDL Validator and have access to it, you are now ready to read and practise chapter 9, *Tutorial: The Autolink Tool*.