Chapter

32

Analyzing TTCN Documents (in Windows)

This chapter contains a reference manual to the Analyzer in the TTCN suite. You can also find information about how to find erroneous tables displayed in the Analyzer log.

The TTCN Browser and TTCN Table Editor are described in chapter 31, *Editing TTCN Documents (in Windows)*.

See chapter 2, *Introduction to the TTCN Suite (in Windows), in the TTCN Suite Getting Started* for an overview of the TTCN suite toolset.

> **Note: Windows version**
>
> **This is the Windows version of the chapter. The UNIX version is chapter 27, *Analyzing TTCN Documents (on UNIX)*.**

Analyzing TTCN Documents

The Analyzer in the TTCN suite does a complete syntax check on both TTCN and ASN.1 (as it is used in TTCN). The Analyzer also performs a number of static semantic checks, mainly the uniqueness and existence of identifiers.

For implementation reasons, the Analyzer does not exactly follow the TTCN standard – some syntax restrictions and relaxations apply. These differences will not affect the vast majority of users, but for reference they are documented together with the standard syntax in chapter 39, *Languages Supported in the TTCN Suite*. This chapter also includes the static semantics supported by the TTCN suite.

During analysis, the TTCN suite compiles an extensive symbol table containing all named objects in the TTCN document and the references between them.

Using the Analyzer

In the TTCN suite, the Analyzer works on the currently active TTCN document. It is also possible to analyze a TTCN document from the Organizer, see "Analyze TTCN" on page 117 in chapter 2, *The Organizer*.

- To analyze a subtree of a TTCN document in the TTCN suite, select a node in the view and then select *Analyze* from the *Build* menu. The selected node and all nodes in its subtree will be analyzed. If the root node is selected, the entire suite will be analyzed.
  – The shortcut is `<F7>`.

- To analyze the entire test suite, regardless of any selections in the test suite, select *Analyze Suite* from the *Build* menu.
  – The shortcut is `<Ctrl+F7>`.

- To analyze an entire modular TTCN system – containing the currently active suite or module – starting from the root suite, select *Analyze System* from the *Build* menu. This is equivalent to analyzing the root node in the Organizer.

  Note that it may take a while before this operation is completed, depending on the size and complexity of the system.

If an item is analyzed incorrectly, this will be recorded as an error message in the log. Items that have been found to be incorrect will be marked in red in all views of the document.

The *Analyzer* Dialog

When you analyze a document for the first time, the *Analyzer* dialog will be opened and you may change the options. Once this has been done, the same options will be used and the dialog will not be opened the next time you start the analysis from a quick-button or a keyboard shortcut. However, the dialog will always be opened when you select an analyzer menu choice. You can also open the dialog by the menu choice *Build* > *Options* or by the shortcut `<Alt+F8>`.

Note that every active TTCN document has its own analysis settings, so in practice, two or more TTCN documents can be edited and/or viewed at the same time in the TTCN suite, and yet have individual settings. But multiple views of the same TTCN document do of course share the same settings.

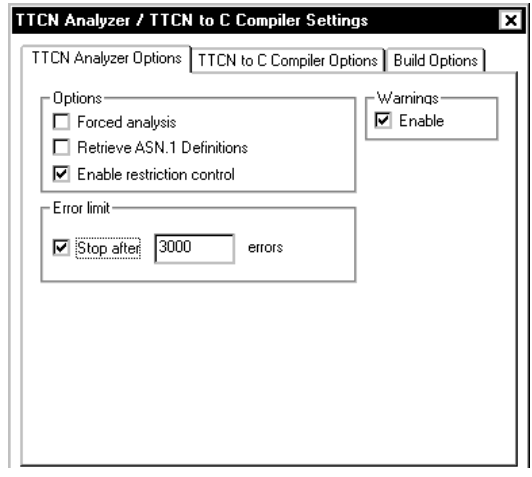

Figure 229: Analyzer dialog

The Analyzer reports all analysis results to the Log Manager. For each different test suite, a new tab will be created to allow the user to easily access the analysis information for different TTCN documents.

Note:

The output of the analysis will not be available if the Log Manager is closed and the option *Auto-Raise Log* is unchecked.

Forced Analysis

Indicates that all selected items are going to be analyzed and no item will be skipped. An item is normally skipped if it is OK.

Retrieve ASN.1 Definitions

Indicates that the definitions of encountered ASN.1 references should be retrieved. An ASN.1 definition is retrieved if this option is set or if the definition has not been fetched before.

Enable Restriction Control

Enables a set of extra controls done on the TTCN code to find even more semantic errors in the analyzed item.

Warnings

Gives more information.

Stop After n Errors

Stops the analysis when the number of errors is reached.

Error Messages

Errors detected during analysis are recorded in the log. For the fields in the header of a table, the error messages have the general format:

```
================================
Table name and table type
Field type 1
Error message
Field type 2
Error message
:
:
Field type n
Error message
================================
```

Example 203 ----------

An error found in the default name (identifier) field of the default dynamic behaviour called UT_DEFAULT:

```
==========================================
Analysis messages in table: UT_DEFAULT of type:
Default Dynamic Behaviour
 Default Name:
The referenced identifier UPPER_PCO is not declared.
UT_DEFAULT (p : UPPER_PCO)
---------------^
==========================================
```

In the body of a table (where the field type name does not uniquely define a field) the row number is also included:

- Row number
- Body type
- Error message

Example 204 ----------

Errors found in a test step table header (the default field) and in the body (rows 1 and 2 of the behaviour description):

```
==========================================
Analysis messages in table: ESTABLISH_CONNECTION of
type: Test Step Dynamic Behaviour
 Default:
Mismatched number of parameters: is 1, should be 0.
   Row: (#1)
      Behaviour Description:
The referenced identifier CR is of wrong type.
UPPER_PCO ! CR
-----------^
   Row: (#2)
      Behaviour Description:
The referenced identifier CC is of wrong type.
UPPER_PCO ? CC
-----------^
==========================================
```

Resolving Forward References

During analysis, the TTCN suite does two main tasks:

- It checks the **syntax** of the abstract TTCN document.
- It checks some of the **static semantics** of the TTCN document.

The syntax of TTCN is defined in ISO/IEC 9646-3 Annex A as a list of BNF productions or rules. See also "The TTCN-MP Syntax Productions in BNF" on page 1530 in chapter 39, *Languages Supported in the TTCN Suite*. The TTCN suite checks all these rules except for a very few minor deviations; for more information, see "The Restrictions in the TTCN Suite" on page 1526 in chapter 39, *Languages Supported in the TTCN Suite*. If a rule is not followed in the TTCN document then this will be reported as an error by the Analyzer.

The static semantic rules of TTCN are also defined in Annex A of ISO/IEC 9646. These are in the form of text statements which *limit* the use of some of the syntax rules, usually in a specific context. "TTCN Static Semantics" on page 1559 and "ASN.1 Static Semantics" on page 1575 in chapter 39, *Languages Supported in the TTCN Suite* describe the static semantics which are controlled by the Analyzer. The static semantics currently supported are mainly of the following types:

- Identifiers (names) are checked for uniqueness with respect to the scoping rules defined in ISO/IEC 9646-3
- The existence of referenced TTCN objects (e.g. a test step) from another TTCN object (e.g. test case) is verified
- Type control and type restriction control

In order to achieve this, the Analyzer constructs a symbol table of all the named TTCN objects that it finds during analysis, e.g. variable names, type identifiers, tables, etc.

It is important to note that TTCN allows *forward reference*s. For example a test step may attach another test step that is declared **later** on in the TTCN document, that is, the referenced (i.e. attached) test step, appears **after** the test step that has done the attach.

The Analyzer has a feature called *back-pass* designed to resolve forward references. The back-pass is automatically invoked at certain points in the analysis process.

Forward references usually appear in the context of ASP/PDU definitions, constraints and as the result of attachment statements in behaviour trees. The reason for this is that the Analyzer does not traverse the selected items in exactly the listed order. The Analyzer order deviates from the listed order in three places:

- Items in the PDU type definitions sub-tree are analyzed before items in the ASP type definitions sub-tree.
- Items in the PDU constraint declarations sub-tree are analyzed before items in the ASP constraint declarations sub-tree.
- Items in the defaults library sub-tree are analyzed before items in the test step library sub-tree and those are analyzed before items in the test cases sub-tree.

ASN.1 External Type/Value References

References to ASN.1 definitions in external ASN.1 modules can occur in the following four TTCN tables:

- Test suite constant declarations by reference
- ASN.1 type definitions by reference
- ASN.1 ASP type definitions by reference
- ASN.1 PDU type definitions by reference

In these tables there are three important fields to consider:

- *Module Identifier* (set by user)
  The name of the ASN.1 module. This should be the real name of the model, i.e. the name stated inside the module itself.
- *Type Reference/Value Reference* (set by user)
  This is the name of the desired type (or value).
- *Type Definition/Value* (set automatically when fetched)
  This is the field where the fetched type definition/value is copied into.

The term *definition* is used below both to denote an ASN.1 type definition and an ASN.1 value.

When an ASN.1 reference is analyzed, the referred definition must be available to the Analyzer. In the pro formas for this kind of reference in the TTCN suite, there is an external column where both the definition and the parse tree is stored.

If the Analyzer encounters any reference to a type/value, the definition is fetched, given the type/value name and the module identifier, and copied into the external field before the field is analyzed. The operation of fetching the definition of the reference is controlled by an option in the Analyzer dialog. Important to note is that the fields *Module Identifier* and *Type/Value Reference* are not parsed. This means that if a change is made in either of these two fields, the row will not be analyzed and thus no fetching will be made. The solution to this is to set the *Enable Forced Analysis* in the Analyzer dialog when altering a reference. See "The Analyzer Dialog" on page 1275.

Analyzing ASN.1 References

This is the general algorithm when encountering an ASN.1 in the analysis phase.

1. First of all, an attempted fetch is only made if no current definition exists or the Analyzer fetching-option is selected.
2. The actual fetching is made, see "Retrieving external ASN.1 Definitions" on page 1280.
3. If the definition is successfully fetched, the identifiers in the definition is converted to TTCN compatible syntax, see "Syntactic Conversion" on page 1281.
4. Finally a possible update of the definition is made and the analysis is continued.

Retrieving external ASN.1 Definitions

From the ASN.1 module a simple parse tree is built to access its contents. The parse tree is (re-)built when either it has not been built before or the ASN.1 module has been modified since last access. The parse tree for an ASN.1 module "lives" for the duration of the whole analysis phase.

Syntactic Conversion

The received definition must be manipulated in the following sense. All identifiers must be syntactically checked and altered if they include any characters that is disallowed in TTCN. The rule is that all '-' (dash) characters in an ASN.1 name are replaced by a '_' (underscore).

No special care has to be considered to ASN.1 comments since they are ignored when the parse tree is built from the module.

External type references in the type definition (e.g "... Module.Type...") will only be converted like above and left for the Analyzer to deal with (see also "Restrictions" on page 1281).

Restrictions

The following features are not supported in TTCN:

- External type reference identifiers on the form *ModuleIdentifier.Type/Valuereference* are not supported.
- Type/value references within the same ASN.1 module are not supported.

Furthermore, there are restrictions in the ASN Utilities when it is used to extract the external types and values. These restrictions are introduced in translating ASN.1 to SDL but since the same algorithm is used for extracting types and values, the restrictions are also relevant for the TTCN suite (see "Translation of ASN.1 to SDL" on page 700 in chapter 14, *The ASN.1 Utilities*).

Error Handling

Any error that occurs while finding, parsing or accessing the ASN.1 module will cause the old definition to remain. Only when a completely successful fetching has been made, the definition will be updated. Possible errors could be:

- The ASN.1 module not found among the dependencies.
- The ASN.1 module file could not be found or was not readable.
- The ASN.1 module file was not syntactically correct.
- The referred type/value could not be found in the module.

See chapter 14, *The ASN.1 Utilities* for a more general view on ASN.1 usage.

TTCN Static Type Restriction Control

The TTCN standard defines a set of semantic rules for type definitions. In particular it defines simple types as types which might contain a true subset of the values which the parent type might contain. The Analyzer now analyzes the TTCN type system and also reports violations of these rules. This analysis is implemented as a post-pass over the related tables, and is only applied when the previous passes have been successful.

Furthermore, restriction control is performed on a number of other constructs in the TTCN language, thus facilitating the task of writing semantically correct TTCN documents. Most of the TTCN types and values are checked and thereby reducing the chance of programming errors slipping through to other tools which depends upon the correctness of the TTCN document.

Application of Static Type Restriction Control

Static type restriction control is applied to at least the following TTCN tables and fields:

- Simple type definitions.
- TTCN structured type / ASP / PDU / CM definitions.
- Test suite constant declarations.
- Test suite variable declarations.
- TTCN structured type / ASP / PDU / CM constraint declarations.
- Actual constraint reference parameter lists.
- Actual parameter lists.

The static type restriction control also evaluates expressions in advance, where possible, and checks that the result does not violate the type restrictions of the field where the value is found. It operates on both types and values, regarding specific values as special cases of types.

Finding Tables from the Analyzer Log

When you are debugging a TTCN document, it may be useful to search for and display named tables. The search will include the entire current TTCN document – no selections are required.

To find an erroneous table identified in the Analyzer log:

1. Place the cursor on the name of the table in the log and then <Ctrl>-right-click the name of the table.
2. In the popup menu that will be opened, select the table name (at the top of the menu), and the table will be opened.

You can also use the Finder to find a table. See "Finding and Sorting Tables" on page 1260 in chapter 31, *Editing TTCN Documents (in Windows)*.