XTC

Language Specification

Version 2.4

Release:	Version 2.4
Date:	August 25, 2016
Status:	Release
Reference:	xtc-specification

This specification defines the XML Timing Cookie (XTC) Language, the interchange format used initially in the Interest project and now updated in the Interested project. This specification defines XTC 2.4.

XTC 2.4 is an XML application conforming to the Extensible Markup Language (XML) 1.0 (Fourth Edition).

Contents

1 What is XTC?

The XML Timing Cookie (XTC) language is an XML application conforming to the Extensible Markup Language 1.0 (Fourth Edition). It is an interchange format that was first used in the Interest project to transport information, analysis requests, and results between the applications for generating or analyzing different parts of embedded software.

An XTC document is made up of two parts: a *common section* and a *cookie section*. The *common section* has a specified format (see Section 2.1). It contains information on the project to be analyzed, information needed for the requested analyses, and the analysis requests as well as analysis results. The *cookie section* (as presented in Section 2.2) can be used by a tool to store tool-specific data that might be useful for restarting an analysis.

1.1 Abstract Model of the Real World

This section describes how real-life projects are modeled in XTC along with the terms that are used.

A system consists of several computation nodes that perform actual tasks. These nodes are called **CPU**s. A CPU is described mainly by its type (architecture), its computation speed, and the program it executes [1].

Each CPU usually executes several tasks that run concurrently. Each task typically consists of a sequence of processes or runnables (AUTOSAR notation). Each runnable or process in turn typically consists of one or more functions. Thus, the entry point of a runnable usually corresponds to a known function name, but may also be the address of an arbitrary instruction.

A user can set up his system with different granularity, and may describe it with different level of detail. Therefore we provide the usage of an **executable** that must be typed with a type tag of **function**, **runnable**, or **task**.

Tasks can be invoked in different execution contexts with different timing behavior. For instance, it may happen that a task only initializes some local variables when it is called for the first time, but performs some real work in subsequent calls resulting in higher running times. Thus, knowledge of the current execution context can considerably improve the precision of worst-case execution time analysis. Therefore we introduce a field to record the **mode** a runnable is executed in.

[1] There are surely more characteristics that define an actual CPU but these details are tool-specific and not relevant at this abstraction level.

Given the CPU, executable and mode elements, we can form an analysis **request**. This may be, e. g., a request for worst-case execution time analysis or stack analysis. The tool that reads an XTC file and performs a requested analysis creates a **response** when it has terminated successfully.

2 Normative Definition of XTC 2.4

Each XTC document consists of a mandatory **common** section and optional **cookie** sections that are contained in the global **xtc** element.

The common section contains analysis requests and information that is needed to process them. The cookie sections are a means to store additional information, e. g., information gathered during an analysis that is useful for later reuse. Cookies are linked to an owner application.

2.1 Common Section

The common section contains an optional **general** section and at least one **CPU** section. It has no attributes.

2.1.1 General Section

The general section can be used to assign an ID to the whole XTC file. This is done by setting an attribute *id*.

An optional **description** section can be used to describe the project or settings the analysis requests are aimed at. It has no attributes and contains simple text that is intended to be displayed to the user by a tool that processes the XTC file.

2.1.2 CPU Section

A CPU section describes the hardware a code snippet is executed on. Its *id* attribute has to contain a unique identifier. A *name* attribute can be used to give the user a hint what architecture is used. The *file* attribute can contain the absolute path to the executable containing the code snippet to be analyzed.

The attributes *speed* and *unit* can be used to specify the clock rate of the CPU. The *speed* is a float value. Valid values for *unit* are `MHz`, `kHz`, and `Hz`. The *unit* attribute is optional. If it is missing, `Hz` is assumed.

A CPU section has to contain at least one **executable** section. This section and the required attribute *type* identify what is executed on the CPU. Possible values for this attribute are:

task: A task is an execution path through address space which is controlled by the operating system. It may consist of several runnables or processes that are sequentially executed.

runnable: A runnable or process is a collection of functions that are sequentially executed.

function: A function is the smallest unit for an execution path through address space. It usually corresponds to functions used in program languages.

An additional attribute *name* exists whose value can be displayed to the user to decide where a requested analysis has to be started. If the assembly name of the function starting the task is known to the application creating the request, it can be passed via the *start* attribute. Note that this attribute value has to be a symbol contained in the executable and thereby references a start address in the executable. Each executable section must have a unique identifier present in its *id* attribute.

To distinguish between different execution contexts, an executable section has to contain at least one **mode** section. Each mode section must have a unique identifier present in its *id* attribute. A description of the execution context can be put in the *name* attribute, so that the user can decide which settings are correct for the analysis.

The mode section has to contain at least one **request** element. There may be optional **response** elements matching the **request** elements. A cookie passed from a tool requesting analyses to a tool performing analyses usually does not contain response elements. A cookie passing in the opposite direction after successful analysis will contain response sections containing the analysis results.

2.1.3 Request / Response Section

A request section contains an analysis request. The type of the analysis requested is specified by the *type* attribute. Possible values are:

WCET: Perform a worst-case execution time analysis

BCET: Perform a best-case execution time analysis

BCET-WCET: Perform both a best-case and worst-case execution time analysis

Stack: Perform a stack height analysis

LoopTrace: Perform a loop iteration range analysis

CallTrace: Perform a computed call target analysis

WCRT: Perform a worst-case response time analysis

BCRT: Perform a best-case response time analysis

BCRT-WCRT: Perform both a best-case and worst-case response time analysis

Activation: Provide information about the activation model of an executable

OS-Overhead: Provide information about scheduling overhead of an executable, like activate or terminate task

TimingProfiler: Identify application parts that cause unsatisfactory execution times

Astree: Static analysis to detect run-time errors in source files

The types of the requests in a mode section must be pairwise different.

The **mode** attribute tells the analyzer how a request is to be handled. The following list shows valid values along with a description:

configuration
: Start the analyzer so an expert user may configure settings on the hardware, etc. In this mode no response is generated but cookies are updated.

interactive
: The analyzer is started in a way allowing the user to control and manipulate common analysis settings, e. g., the entry point.

batch
: Try to start the analyzer in batch mode, i. e. running in the background with no need for additional user interaction. This is only possible if all information required for the analysis is present.

If no **mode** attribute is given, *interactive* mode is used.

A request section may contain **option** elements to pass tool-specific information. Each option has a *name* and a *value* attribute. The contents of these attributes are tool-specific (see Section 3 for examples).

For each request element, there may be a corresponding response section. The association of response sections to request sections is done via the type. This is possible since the types of the requests in a mode section must be pairwise different.

- A response section of type WCET contains a single **WCET** element. A **WCET** element has two attributes:
  - The *value* attribute contains a float number denoting the time spent on the worst-case path.
  - The *unit* attribute may be set to `us` (microseconds), `ms` (milliseconds), `s` (seconds), or `cycles` (CPU cycles). Default is `cycles`.

  A **WCET** element may additionally contain an **UCB** element. This element is available if a UCB analysis has been requested initially (see Section 4). An **UCB** element has four attributes:
  - The *maxNumber* contains the maximum number of useful cache blocks (UCBs).
  - The *blockPenalty* attribute contains a float number denoting the eviction costs of a single UCB.
  - The *totalPenalty* attribute contains a float number denoting the eviction costs of the maximum number of UCBs.
  - The *unit* attribute may be set to `us` (microseconds), `ms` (milliseconds), `s` (seconds), or `cycles` (CPU cycles). Default is `cycles`.
- A response section of type BCET contains a single **BCET** element. A **BCET** element has two attributes:
  - The *value* attribute contains a float number denoting the time spent on the worst-case path.
  - The *unit* attribute may be set to `us` (microseconds), `ms` (milliseconds), `s` (seconds), or `cycles` (CPU cycles). Default is `cycles`.
- A response section of type Stack may contain one or two **stack-usage** elements, depending on the number of stacks in the target architecture (some targets distinguish between a system stack and a user stack). A **stack-usage** element has four attributes:
  - The *type* attribute indicates the type of the stack. It may be `System` or `User`.
  - The attributes *min* and *max* contain the lower end and the upper end of the result interval computed by stack analysis. Their values may be integers or the special values `BOT` and `TOP`.
  - The *unit* attribute may hold an integer between 1 and 127. It represents the size of

the unit used for *min* and *max*, measured in bytes. For instance, if *max* is 10 and *unit* is 4, this means that the maximum stack usage is 10 units of 4 bytes each, i. e. 40 bytes. The default value for *unit* is 1, which means that *max* is the maximum stack usage measured in bytes.

- A response section of type TimingProfiler contains a single **timing-profiler** element. A **timing-profiler** element has two attributes:
  - The *value* attribute contains a float number denoting the time spent on the worst-case path.
  - The *unit* attribute may be set to `us` (microseconds), `ms` (milliseconds), `s` (seconds), or `cycles` (CPU cycles). Default is `cycles`.
- A response section of type Astrée contains a single instance of each of the following elements:
  - The *analysis_id* holds the id of the analysis created on the server to process the request.
  - The *success* element indicates whether the analysis run was successful.
  - The *errors* element indicates the number of errors reported by the analyzer.
  - The *alarms* element indicates the number of alarms reported by the analyzer.
  - The *coverage* element indicates the total coverage for the project.
- A response section of type LoopTrace contains for each loop a **LoopTrace** element that has four attributes:
  - The *min* attribute indicates the minimum loop iteration count.
  - The *max* attribute indicates the maximum loop iteration count.
  - The *start_address* attribute to identify an instruction before the loop.
  - The *count_address* attribute to identify the body of the loop.
- A response section of type CallTrace contains for requested computed call a **CallTrace** element that may contain an arbitrary number of **target** elements. Both the **CallTrace** and the **target** element feature a single *address* attribute. For the **CallTrace** the *address* denotes the addresses of the program point where the computed call resides. The *ad-*

dress attribute of the **target** element indicates the target address of the corresponding computed call.

- A response section of type Activation may contain any event model type (StandardEventmodelType, PatternEventmodelType or InterruptEventmodelType).
  - *StandardEventModel* consists of the three values *period*, *jitter* and *mindist*. An additional field indicates to indicate that this event is sporadic.
  - The *PatternEventModel* has similar attributes as the StandardEventModel. It additionally has the possibility to describe an event sequence. While defining the StandardEventModel it was implicit that an event would always occur at the beginning of the period. While defining a *PatternEventModel* each such activation has to be specified.
  - The *InterruptEventModel* defines a sequence of *eventPairs*. Each pair defines the minimum and maximum distance between events. The first pair in the sequence defines the distances between two events. The second pair in the sequence defines the distances between three events, and so forth.
- The *OsOverheadTypeContent* provides information regarding the overhead induced to the selected element by the OS scheduling. These overheads are detailed below.
  - The *activation* overhead is the time required by the OS to schedule the activation of the element.
  - The *termination* overhead is the time required by the OS to clear up after the element has finished execution.
  - The *contextSwitch* overhead is the time it takes for the OS to preempt the selected element and switch the execution to another task or ISR.
  - The *contextSwitchCachePenalty* is the largest execution time effect on the element if its cache is destroyed, at any point, during execution. This value is used during scheduling analysis to compensate for the undisturbed execution assumption during static WCET analysis.

2.2 Cookie Section

Each cookie section is owned by a single tool. Its name has to be specified in the *owner* attribute. Currently used values are:

alauncher: AbsInt's tools (aiT worst-case execution time analyzer, StackAnalyzer, TimingProfiler, Astrée Analyzer for static run-time error analysis)

RT-Druid: Evidence's configuration and schedulability analysis tool

SymTA/S: Symtavision's SymTA/S tool suite

Each tool can only have one cookie section.

The organization of a cookie section is defined by the owner application. The XTC specification only requires a unique namespace.

3 Example XTC Requests

3.1 UCB Analysis via XTC

Example demonstrating how to trigger the UCB analysis via XTC:

- Input XTC cookie enabling the UCB analysis:

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  <common>
    <CPU speed="120.0" unit="MHz" id="ID_0" name="mpc5554" file="edn.elf">
      <executable type="runnable" id="ID_1" name="edn" start="main">
        <mode id="ID_2">
          <description>EDN</description>
          <request mode="interactive" vendor="SymTA/S" type="WCET">
            <option value="mpc55xx" name="a3:cpu"/>
            <option value="edn.ais" name="a3:global_ais_file"/>
            <option value="edn.msf" name="a3:machine_settings_file"/>
            <option value="true" name="a3:ucb_analysis"/>
            <option value="3" name="a3:ucb_penalty"/>
          </request>
        </mode>
      </executable>
    </CPU>
  </common>
</xtc>
```

- Final XTC cookie with UCB analysis results:

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  <common>
    <CPU speed="120.0" unit="MHz" id="ID_0" name="mpc5554" file="edn.elf">
      <executable type="runnable" id="ID_1" name="edn" start="main">
        <mode id="ID_2">
          <description>EDN</description>
          <request mode="interactive" vendor="SymTA/S" type="WCET">
            <option value="mpc55xx" name="a3:cpu"/>
            <option value="edn.ais" name="a3:global_ais_file"/>
            <option value="edn.msf" name="a3:machine_settings_file"/>
            <option value="true" name="a3:ucb_analysis"/>
            <option value="3" name="a3:ucb_penalty"/>
          </request>
          <response vendor="a3" type="WCET">
            <WCET unit="us" value="856.733333">
              <UCB totalPenalty="0.250000"
                   unit="us"
                   blockPenalty="0.025000"
                   maxNumber="10"/>
            </WCET>
```

```
          </response>
        </mode>
      </executable>
    </CPU>
  </common>
</xtc>
```

3.2 Determination of Unknown Loop Bounds

Tool interaction between SymTA/S, T1 and aiT (unknown loop bounds):

- Input XTC cookie from SymTA/S to compute the WCET of some entry point:

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  <common>
    <CPU speed="56" unit="MHz" id="ID_0" name="ECU" file="../temp.elf">
      <executable type="runnable" id="ID_1" name="Main" start="MAIN_LOOP">
        <mode id="ID_2">
          <description>Main loop.</description>
          <request mode="interactive" vendor="SymTA/S" type="WCET">
            <option value="../hardware.msf"
                    name="a3:machine_settings_file"/>
          </request>
        </mode>
      </executable>
    </CPU>
  </common>
</xtc>
```

- Intermediate XTC cookie featuring a LoopTrace request (only modified parts are shown):

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  ...
        <mode id="ID_2">
          ...
          <request vendor="a3" type="LoopTrace" source="analysis">
            <option value="0x43691c" name="T1:start_addresses"/>
            <option value="0x436920" name="T1:count_addresses"/>
            <option value="0x436930" name="T1:end_addresses"/>
          </request>
          ...
```

- Response from T1 featuring measured loop bounds (only modified parts are shown):

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  ...
        <mode id="ID_2">
```

```
...
<response type="LoopTrace" vendor="T1">
    <LoopTrace start_address="0x0043691c"
               count_address="0x00436920"
               min="80"
               max="100"/>
</response>
...
```

- Final XTC cookie featuring WCET analysis results

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  <common>
    <CPU speed="56" unit="MHz" id="ID_0" name="ECU" file="../temp.elf">
      <executable type="runnable" id="ID_1" name="Main" start="MAIN_LOOP">
        <mode id="ID_2">
          <description>Main loop.</description>
          <request mode="interactive" vendor="SymTA/S" type="WCET">
            <option value="../hardware.msf"
                    name="a3:machine_settings_file"/>
          </request>
          <request vendor="a3" type="LoopTrace" source="analysis">
            <option value="0x43691c" name="T1:start_addresses"/>
            <option value="0x436920" name="T1:count_addresses"/>
            <option value="0x436930" name="T1:end_addresses"/>
          </request>
          <response type="LoopTrace" vendor="T1">
              <LoopTrace start_address="0x0043691c"
                         count_address="0x00436920"
                         min="80"
                         max="100"/>
          </response>
          <response vendor="a3" type="WCET">
            <WCET unit="us" value="20.732143"/>
          </response>
        </mode>
      </executable>
    </CPU>
  </common>
</xtc>
```

3.3 Resolving of Unresolved Computed Calls

Tool interaction between SymTA/S, T1 and aiT (unresolved computed calls):

- Input XTC cookie from SymTA/S to compute the WCET of some entry point:

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  <common>
    <CPU speed="56" unit="MHz" id="ID_0" name="ECU" file="../temp.elf">
      <executable type="runnable" id="ID_1" name="Main" start="MAIN_LOOP">
        <mode id="ID_2">
          <description>Main loop.</description>
          <request mode="interactive" vendor="SymTA/S" type="WCET">
            <option value="../hardware.msf"
                    name="a3:machine_settings_file"/>
          </request>
```

```
        </mode>
      </executable>
    </CPU>
  </common>
</xtc>
```

- Intermediate XTC cookie featuring a CallTrace request (only modified parts are shown):

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  ...
        <mode id="ID_2">
          ...
          <request vendor="a3" type="CallTrace">
            <option value="0x41b5f4,0x41ba04" name="T1:call_addresses"/>
          </request>
          ...
```

- Response from T1 featuring measured computed call targets (only modified parts are shown):

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  ...
        <mode id="ID_2">
          ...
          <response type="CallTrace" vendor="T1">
            <CallTrace address="0x0041B5F4">
              <target address="0x0042D2CC"/>
            </CallTrace>
            <CallTrace address="0x0041BA04"/>
          </response>
          ...
```

- Final XTC cookie featuring WCET analysis results

```
<xtc version="2.0" schemaLocation="http://www.absint.com/xtc xtc.xsd">
  <common>
    <CPU speed="56" unit="MHz" id="ID_0" name="ECU" file="../temp.elf">
      <executable type="runnable" id="ID_1" name="Main" start="MAIN_LOOP">
        <mode id="ID_2">
          <description>Main loop.</description>
          <request mode="interactive" vendor="SymTA/S" type="WCET">
            <option value="../hardware.msf"
                    name="a3:machine_settings_file"/>
          </request>
          <request vendor="a3" type="CallTrace">
            <option value="0x41b5f4,0x41ba04" name="T1:call_addresses"/>
          </request>
          <response type="CallTrace" vendor="T1">
            <CallTrace address="0x0041B5F4">
              <target address="0x0042D2CC"/>
            </CallTrace>
```

```
            <CallTrace address="0x0041BA04"/>
          </response>
          <response vendor="a3" type="WCET">
            <WCET unit="us" value="16.643"/>
          </response>
        </mode>
      </executable>
    </CPU>
  </common>
</xtc>
```

3.4 Several Stack and WCET Analysis

Complex XTC request file featuring several WCET-analysis and stack-analysis requests:

```
<xtc
     xmlns:a3="http://www.all-times.org/xtc/a3"
     xmlns:SymTA-S="http://www.all-times.org/xtc/SymTA-S"
     xmlns:RTDruid="http://www.all-times.org/xtc/RTDruid"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <common>
    <general id="id1">
      <description>Description of the project</description>
    </general>
    <CPU id="id2" name="Motorola PowerPC 565">
      <executable type="runnable" id="id3" name="Main entry" start="_main">
        <mode id="id4">
          <description>Specify mode for the given entry point</description>
          <request type="WCET" mode="interactive">
            <option name="a3:cpu" value="mpc5xx"/>
            <option name="a3:build" value="55383"/>
            <option name="a3:target" value="MPC565"/>
            <option name="a3:global_ais_file" value="common.ais"/>
            <option name="a3:ais_file" value="main.ais"/>
            <option name="a3:gdl_file" value="graph.gdl"/>
            <option name="a3:xml_report_file" value="report-wcet.xml"/>
            <option name="a3:stack_address" value="0x3ffff8"/>
          </request>
          <request type="Stack" mode="interactive">
            <option name="a3:cpu" value="mpc5xx"/>
            <option name="a3:build" value="55383"/>
            <option name="a3:target" value="MPC565"/>
            <option name="a3:ais_file" value="test.ais"/>
            <option name="a3:xml_report_file" value="report-stack.xml"/>
            <option name="a3:stack_address" value="0x3ffff8"/>
          </request>
        </mode>
      </executable>
      <executable type="runnable" id="id5" name="Computation" start="_compute">
        <mode id="id6">
          <description>Specify mode for the given entry point</description>
          <request type="WCET" mode="interactive">
            <option name="a3:cpu" value="mpc5xx"/>
            <option name="a3:build" value="55383"/>
            <option name="a3:target" value="MPC565"/>
            <option name="a3:ais_file" value="compute.ais"/>
            <option name="a3:xml_report_file" value="report.xml"/>
            <option name="a3:stack_address" value="0x3ffff0"/>
          </request>
        </mode>
      </executable>
    </CPU>
```

```
  </common>
</xtc>
```

4 Options used by aiT, StackAnalyzer and TimingProfiler

This section lists the name attributes of a request's **option** element that are recognized by aiT, StackAnalyzer and TimingProfiler, and describes how the value is used for the analysis. They use the prefix **a3** because aiT, StackAnalyzer and TimingProfiler are invoked via the a3 binary. If the value of an option is a relative path or a relative file name then the value is interpreted relative to the location of the XTC document.

- **a3:cpu**
  The name of the CPU so a suitable member of the aiT/StackAnalyzer family can be started.

- **a3:build**
  The build number of aiT/StackAnalyzer to use.

- **a3:target**
  If there are several cores (i. e. targets) to choose from, this option can be used to select the desired one. For example, a3 for the MPC5xx CPU supports targets such as MPC555, MPC565, etc.

- **a3:global_ais_file**
  The name of the AIS file that is used for all analyses.

- **a3:ais_file**
  The name of the AIS file that is used for the analysis in addition to the global AIS file.

- **a3:gdl_file**
  The name of the file to be used for visualizing the analysis results.

- **a3:report_file**
  The name of a plain text report file.

- **a3:xml_report_file**
  The name of a XML report file.

- **a3:xml_show_per_context_info**
  Output information in the XML report per context.

- **a3:html_report_file**
  The name of an HTML report summary file that should be created from the XML report file.

- **a3:stack_address**
  The initial value of the stack pointer.

- **a3:scade_xml_file**
  The name of the XML file that is needed to launch aiT/StackAnalyzer if bundled with SCADE.

- **a3:machine_settings_file**
  Instructs aiT/StackAnalyzer to read machine settings from this file if supported by the target.

- **a3:ucb_analysis**
  Instructs aiT to perform a useful cache block (UCB) analysis if supported by the target. This is only valid for a WCET analysis.

- **a3:ucb_penalty**
  The eviction costs for a useful cache block. If not specified, one CPU cycle is used as the default.

- **a3:data_dictionary**
  The name of the data dictionary XML file in extended format that is used to extract additional information.

- **a3:includes**
  A list of paths that are used as include paths when preprocessing source files.

- **a3:skip_wcet_main_entry**
  If additional starts are found during analysis, do not perform a WCET analysis for the specified main entry.

- **a3:stack_analysis_mode**
  Set the stack analysis mode. Possible values are:

  - `normal`,

  - `optimized_with_contexts` and

  - `optimized_without_contexts`.

5 Options used by Astrée

This section lists the name attributes of a request's **option** element that are recognized by Astrée and describes how the value is used for the analysis.

If the value of an option is a relative path or a relative file name then the value is interpreted relative to the location of the XTC document.

- **astree:build**
  The build number of Astrée to use.

- **astree:server**
  The Astrée server to connect to.

- **astree:port**
  The port on which the Astrée server is listening.

- **astree:dax**
  Specifies a DAX file describing the analysis to be performed. All other options are ignored.

- **astree:source_files**
  A list of C source files that are to be analyzed but need to be preprocessed first.

- **astree:preprocessed_source_files**
  A list of C source files that are to be analyzed and are already preprocessed.

- **astree:includes**
  A list of paths that are used as include paths when preprocessing source files.

- **astree:defines**
  A list of paths that are used as defines paths when preprocessing source files. Each define entry in the list has the form `<name>[=<value>]`.

- **astree:standard_c_library_stubs** Whether stubs for the standard C library should be used for the analysis.

- **astree:keep_comments**
  Whether comments should be kept in the preprocessed source files.

- **astree:analysis_start**
  The name of the function that is the start of the analysis. This option overwrites the

start attribute of the **executable** element.

- **astree:wrapper**
  The name of file containing the wrapper function used as analysis start.

- **astree:data_dictionary**
  The name of the XML data dictionary as generated from TargetLink in extended format.

- **astree:data_dictionary_subsystems**
  List of subsystems for which information should be extracted from the data dictionary.

- **astree:data_dictionary_toplevel_subsystems**
  List of subsystems declared to be top-level, i.e., for which function calls to their step functions are to be generated within the analysis wrapper.

- **astree:data_dictionary_flags**
  List of flags that specify what kind of information should be extracted from the data dictionary and used for the analysis. Valid flags are:

  - `ranges`
    generate annotations for the ranges of input and output variables of funtions

  - `parameters=mode` where `mode` is one of `range` and `volatile-range`
    generate annotations for the ranges of calibration parameters

  - `array-parameters`
    also generate annotations for calibration parameters that are arrays; no effect if `parameters` is not set

  - `lookup`
    generate annotations to improve the precision of of look-up and interpolation routines

  - `stubs`
    substitute the look-up and interpolation routines by stubs

- **astree:postprocess_script**
  The name of the executable that is used to postprocess wrapper and annotation files when converting data dictionaries.

- **astree:analysis_options**
  The name of the file containing analysis options.

- **astree:analysis_abi**

The name of the file defining the application binary interface.

- **astree:analysis_annotations**
  The name of the file containing the annotations to use for the analysis in AAL.

- **astree:log_file**
  The name of the textual analysis log file.

- **astree:xml_log_file**
  The name of the XML file containing all relevant information of the analysis run in a form that is convenient for machine processing.

- **astree:xml_summary_file**
  The name of the XML summary file.

- **astree:xml_coverage_file**
  The name of the XML coverage file.

- **astree:xml_alarms_file**
  The name of the XML alarms file ordered by occurrence.

- **astree:xml_alarms_by_type_file**
  The name of the XML alarms file ordered by type.

- **astree:xml_alarms_by_source_file**
  The name of the XML alarms file ordered by source file.

- **astree:timeout**
  Timeout in seconds after which the analysis is stopped. Only effective in batch mode.

Appendix: Document History

This appendix describes the revision history of this document. Dates are written in the format DD-MM-YYYY. A *document release* is a release version of the document. Document releases can be identified by the date occurring on the title page. The history given below lists all changes to this document.

16-08-2016	Updated examples.
11-03-2016	Added support for a3 option *stack_analysis_mode*.
12-01-2016	Removed references to old aiT options.
18-12-2015	Added support for Astrée option *dax*.
18-09-2015	Added support for Astrée option *data_dictionary_toplevel_subsystems*.
17-06-2015	Adder support for Astrée option *filter*.
03-04-2014	Renamed owner from a3 to alauncher.
09-10-2013	Added support for xml_show_per_context_info option.
06-08-2013	Removed support for xml_result_file.
02-08-2013	Added support for html_report_file optionastree option, removed xml_style_sheet option.
17-06-2013	Added support for Astrée option *analysis_annotations*.
02-05-2013	Added support for Astrée option *data_dictionary_subsystems*.
06-12-2012	Replaced options *volatile* and *const_volatile* by *parameters* and *array_parameters*.
20-09-2012	Added support for additional aiT options xml_style_sheet.
07-08-2012	Changed the format of the flags for the Astrée data_dictionary_flags option and described the response section for Astrée.
26-01-2012	Added support for Astrée data_dictionary_flags option and an option to keep comments.
06-01-2012	Added support for additional aiT options.
17-11-2011	Added support for Astrée analysis requests and the data dictionary option.
03-05-2011	Added TimingExplorer request type.
13-01-2011	Added target option.
28-04-2010	Added CallTrace response and requests.
15-02-2010	Added some more XTC cookie examples.
05-01-2010	Added documentation on how to trigger a UCB analysis.
07-09-2009	Adapted document to XTC version 2.0.
02-02-2009	Added machine settings file option.
30-07-2008	Revised XTC examples and updated description of aiT options.
23-07-2008	renamed top element to *xtc*.

30-06-2008	Update of description of response elements.
	Other minor improvements.
01-04-2008	Added scade_xml_file option description.
	Added global_ais_file option description.
03-03-2008	Added description of mode attribute for a request.
	Added appendix containing options recognized by aiT.
12-11-2007	Correction of typos, and a bit more explanations.
	Addition of an example (Section 3).
07-10-2007	Minor update: Allow *request* items to have a *start* attribute that contains the name of the function to be analyzed.
16-08-2007	First release.
18-06-2007	First draft.