
Glossary

Glossary of Meta-Environment Terminology

Abstract Data Type (ADT)	Structural description of the interface of a component. Used by APIGEN to generate an Application Programmer's Interface (API) for the component.
APIGEN	Application Programming Interface (API) generator. Given a datastructure definition (in the form of an Abstract Data Type), APIGEN generates C or Java code to access that datastructure. Internally, the datastructure is represented as ATerm.
ASF	Algebraic Specification Formalism. This a notation for describing rewrite rules and is mostly used for defining software analysis, fact extraction, and software transformation.
ASF checker	ASF checker performs static checking on ASF definitions.
ASF compiler	The ASF compiler transforms ASF specifications to C code. That C code is compiled and linked with a run-time library. This leads very efficient execution of specifications.
ASF interpreter	The ASF interpreter takes the equation sections from the ASF+SDF specification and applies them to the parsed input term. It produces another parse tree as output.
ASF operations	ASF operations provides all operations to read and modify ASF definitions.
ASF+SDF	The combination of the formalisms ASF and SDF. ASF+SDF can describe both the syntax of a language and the operations on that language (checking, execution, analysis, transformation).
ASF+SDF checker	The ASF+SDF checker checks an SDF definition for compatibility for use with ASF, and also checks some production attributes that are specifically interpreted by ASF rewriting engines. It is an extension of the SDF checker.
ASF+SDF library	A collection of elementary datatypes (lists, tables, etc.), language grammars (C, COBOL, Java, SDF, etc.) and utilities.
ASF+SDF modules	The ASF+SDF modules describe: <ul style="list-style-type: none">• the syntax of the base language,• the functions that can be applied to base language programs. Note that the ASF+SDF modules may define several base languages at the same time.
AsFix	ASF+SDF Fixed format. The dataformat used to represent parse trees. AsFix is a specialized view on ATerms. Important features are: <ul style="list-style-type: none">• The AsFix format is a full parse trees that contains all the original layout and comments from the original source code program that was parsed.• The AsFix format is self-descriptive: each subtree contains information about the exact grammar production that has been used to parse the text that has resulted in that parse tree.

	<ul style="list-style-type: none">• The AsFix format does not contain source code coordinates per se, but a separate tool (addPosInfo) can easily compute these coordinates and add them to the parse tree in the form of annotations.
ATerm	<p>Annotated terms. A dataformat used for the internal representation of all data. Distinguishing features are:</p> <ul style="list-style-type: none">• ATerms are language-independent and can be processed by programs in any language.• ATerms can be annotated with auxiliary information that does not affect the tree structure.• ATerms preserve <i>maximal subterm sharing</i>. This means that common parts of the data are not duplicated but shared. This leads to considerable size-reduction of the data.
Box	<p>Intermediate representation of the prettyprinter. A parse tree is first converted to a box term that includes all desired formatting directives (alignment, font and color directives, and the like). Next, the box term is converted to various output formats (plain text, HTML, etc.).</p>
Configuration manager	<p>The configuration manager handles user settings and preferences.</p>
Compiled specification	<p>It is possible to compile the ASF+SDF into a very efficient executable form. Compiled specifications can be used via the command line interface of The Meta-Environment by power users.</p>
Debugger	<p>The debugger allows a step-by-step execution of the rewrite rules defined in ASF+SDF specifications.</p>
Errors & warnings	<p>Errors & warnings pinpoint any errors in the ASF+SDF modules or input terms.</p>
Graph	<p>Data format to represent graphs. Used as representation of import graphs and parse trees that are displayed in the GUI.</p>
Graphical user interface (GUI)	<p>The graphical user-interface (GUI) gives end-users access to the system's functionality. It is a "sovereign" user-interface that occupies the complete desktop window and provides functionality like:</p> <ul style="list-style-type: none">• Opening, editing and closing ASF+SDF modules.• Opening, editing, reducing and closing input terms.• A graphical and tree-structured display of the import relations between ASF+SDF modules.• A graphical display of parse trees.• Error and progress indications.
Location	<p>Data format to describe locations in source code.</p>
Input term	<p>A text that conforms to the syntax defined in the ASF+SDF modules. This includes:</p> <ul style="list-style-type: none">• the syntax of the base language (e.g., C, Java, Boolean expressions, a domain-specific language),• the functions that can be applied to base language programs (e.g., typecheck, extract facts, compile, remove unused methods).

	<p>The input term can be freely edited and is checked for syntactic correctness before any function is applied to it. It is possible to simultaneously edit different input terms in different base languages.</p>
Module manager	<p>The module manager is responsible for all information related to the ASF+SDF modules that reside in the system.</p>
Output term	<p>A text that describes the result of applying a function to a program in the base language. Note that this text conforms to the syntax R, where R is the result sort of the function that was applied. With Java-to-Java transformation the result sort will be Java. When no function is applied, the output term is identical to the input term.</p>
Parser	<p>The parser takes a parse table (as produced by the parse table generator) and text (as provided by a text editor) as input and produces a parse tree as output. Any errors are shown in the error display of the GUI.</p>
Parse table	<p>A parse table is an efficient representation (ATerm) of the base language as defined by the ASF+SDF modules and enables efficient parsing.</p>
Parse table generator	<p>The parse table generator takes syntax sections from the ASF+SDF specification and converts them to a parse table to be used for the parsing of terms.</p>
Parse tree	<p>Tree-structured representation (in AsFix) of a text that has been analyzed by a parser.</p>
Prettyprinter	<p>The prettyprinter converts parse trees to text. The prettyprinter uses default rules to insert layout in a parse tree so that its corresponding text is presented in a uniform way. Optionally, the ASF+SDF specification may contain formatting rules that can replace this default behaviour.</p>
Rscript	<p>A small scripting language for defining relational expressions. Used for the analysis of facts extracted from software.</p>
SDF	<p>Syntax Definition Formalism. A notation for describing the grammar of programming and application languages.</p>
SDF checker	<p>SDF checker performs static checking on SDF definitions.</p>
SDF operations	<p>SDF operations provides all operations to read and modify SDF definitions.</p>
Sisyphus	<p>A system for continuous integration that rebuilds the system after each change that is committed by a developer.</p>
Structure editor	<p>A syntax-directed editor that closely cooperates with the text editor. It is mostly used for syntax-directed navigation through the text. The structure-editor does not appear as such in the GUI but all its functionality is visible through the text editor.</p>
Summary	<p>An error or message summary. A dataformat for the internal representation of errors and messages. Summaries are produced by checker and compilers and are used by the GUI.</p>
Term store	<p>The term store contains all parse tables, parse trees and other intermediate data that is generated during execution of The Meta-Environment. This includes:</p> <ul style="list-style-type: none">• The parse tree for each ASF+SDF module.• The parse tree for each parsed input term.

	<ul style="list-style-type: none">• The parse tree for each generated output term.
Text editor	The text editor allows text editing on ASF+SDF modules and input terms. Multiple editors may be opened; each appears as a tabbed window in one of the panes of the GUI.
The Meta-Environment	The architecture of The Meta-Environment (or just "the system") is the primary object of study of this document.
ToolBus	<p>The ToolBus coordination architecture enables the flexible and controlled combination and orchestration of software components. It is used as backbone for The Meta-Environment. The ToolBus has the following characteristics:</p> <ul style="list-style-type: none">• Components (or <i>tools</i> in ToolBus parlance) can be written in different programming languages.• Components can be running on different machines.• All interactions between components are regulated by a ToolBus script (or Tscript for short) that is executing in the ToolBus. Tscript is a concurrent language that allows the definition of parallel processes, messaging between these processes and interaction between processes and tools.
Tscript	The script that describes the cooperation between components in a ToolBus-based application.