SNIA SMI-S Recipe Interpreter

Jun Feng, Liu
Jin Xin, Ying,
IBM STG Lab, 2009, April

Agenda

- **SMI Recipe Language (SMIRL)**
- SMIRL Interpreter
- Demo
- Interpreter Structure
- Summary

SMI Recipe Language (SMIRL)

Background

- SMI-S specification allows simple, standardized, and cost-effective management of heterogeneous storage environments. The SMI-S allows an IT manager to use an OEM's software to manage third-party storage products and vise versa.

- Since SMI-S is a standard based technology, SMI-S developers need to do compliance testing to make sure their implementations follow the standard's description.

Background

- SMIRL(SMI Recipe Language) is part of SMI-S specification, which was defined to describe the logic of Recipe. SMI-S Recipe is a set of instructions defined for a particular profile, sub-profile, it specifies an interoperable means for accomplishing a particular task across all conformant implementations.

- Each recipe defines an interoperable series of interactions (between a SMI-S Client and a SMI-S Server) required to manage storage devices or applications, and lists the operations required for the CIM Client realize functionality.

- Recipes are expressed as CIM or SLP operations, and have clear syntax definition.

SMIRL Example


```
$Subprofiles[] = Associators (
    $RegisteredProfile->,
    “CIM_SubProfileRequiresProfile”,
    “CIM_RegisteredProfile”,
    NULL, NULL, false, false, NULL)
// Step 3: Verify that each Subprofile has the same version as the
// Profile
for #i in $Subprofiles[]
{
    #SubprofileVersion = $Subprofile[#i].RegisteredVersion
    if (!compare(#SubprofileVersion, #ProfileVersion))
    {
        Error(“Subprofile version mismatch with Profile version”)
    }
}
```

SMIRL Syntax

- SMIRL syntax does not follow any program language, but has clear definition in specification section “Recipe Pseudo Code Conventions”

- **General Syntax**

  <condition> logical statement that evaluates to true (Boolean)

  <EXIT:*success message*> Exits the recipe with a success status code.

  <ERROR! Error condition> Exits the recipe with failure status code.

  …..

SMIRL Syntax

- CIM related variable and methods

  $name represents a single instance (CIMInstance)
  $name. property represents a property in CIMInstance
  $name.getObjectPath()
  $name.getNameSpace()
  $name-> represents an object path (CIMObjectPath)
  $name->[] represents an array of object names
  $name->[].size()

SMIRL Syntax

- Built-in Function
  - boolean = compare(<variable>, <variable>)
  - $instance = newInstance("CIM_Class")
  - boolean =contains(<test value>, <variable array>)

Agenda

- SMI Recipe Language (SMIRL)
- **SMIRL Interpreter**
- Demo
- Interpreter Structure
- Summary

Problem

- As part of the certificate compliance tests, any SMI-S implementation needs to pass the recipe validation. However, SMIRL is just pseudo code, which can not be executed and the syntax was not compliance with any of today's language.
- SMI-S developers have to read the specification and understand the logic of each recipe, then re-write the test code with other executable language like JAVA or C++.
- Today’s validation tools like CTP require manually coding the recipe logical into JAVA to implement the test.
- With the evolution of the SMI-S specification, test client have to change from version to version to capture the latest changes of specification. This requires a lot extra development effort and error prone.

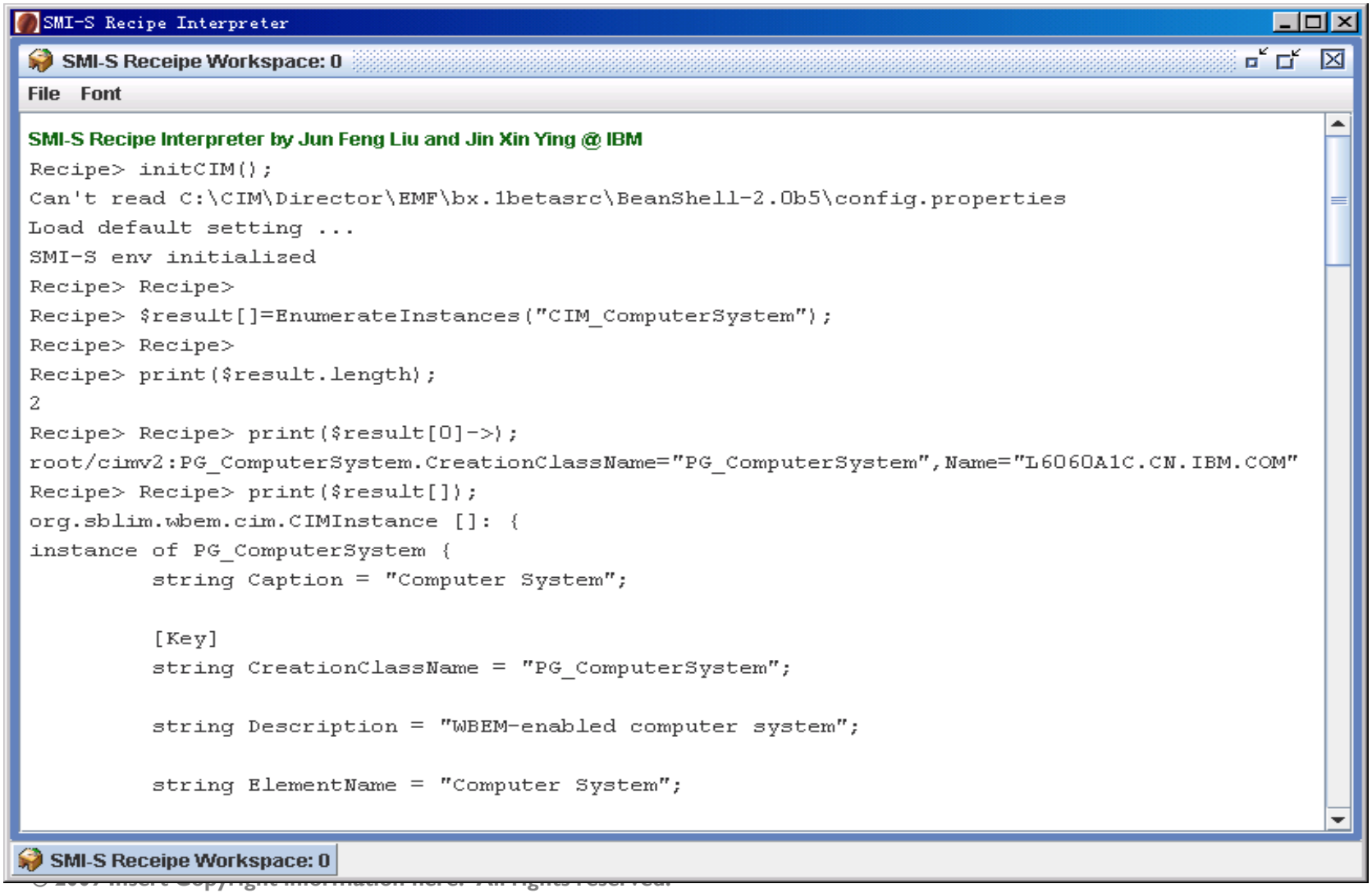
SMI-S Recipe Interpreter

- A recipe interpreter is provided which can understand, and execute recipe “language” directly.
- The tool has log file to record each work steps, so that user can understand what kind of in-compatibility problem they may have.
- The tool also generate validation points from Recipe. User can use the configuration to control the validate level.

Agenda

- SMI Recipe Language (SMIRL)
- SMIRL Interpreter
- **Demo**
- Interpreter Structure
- Summary

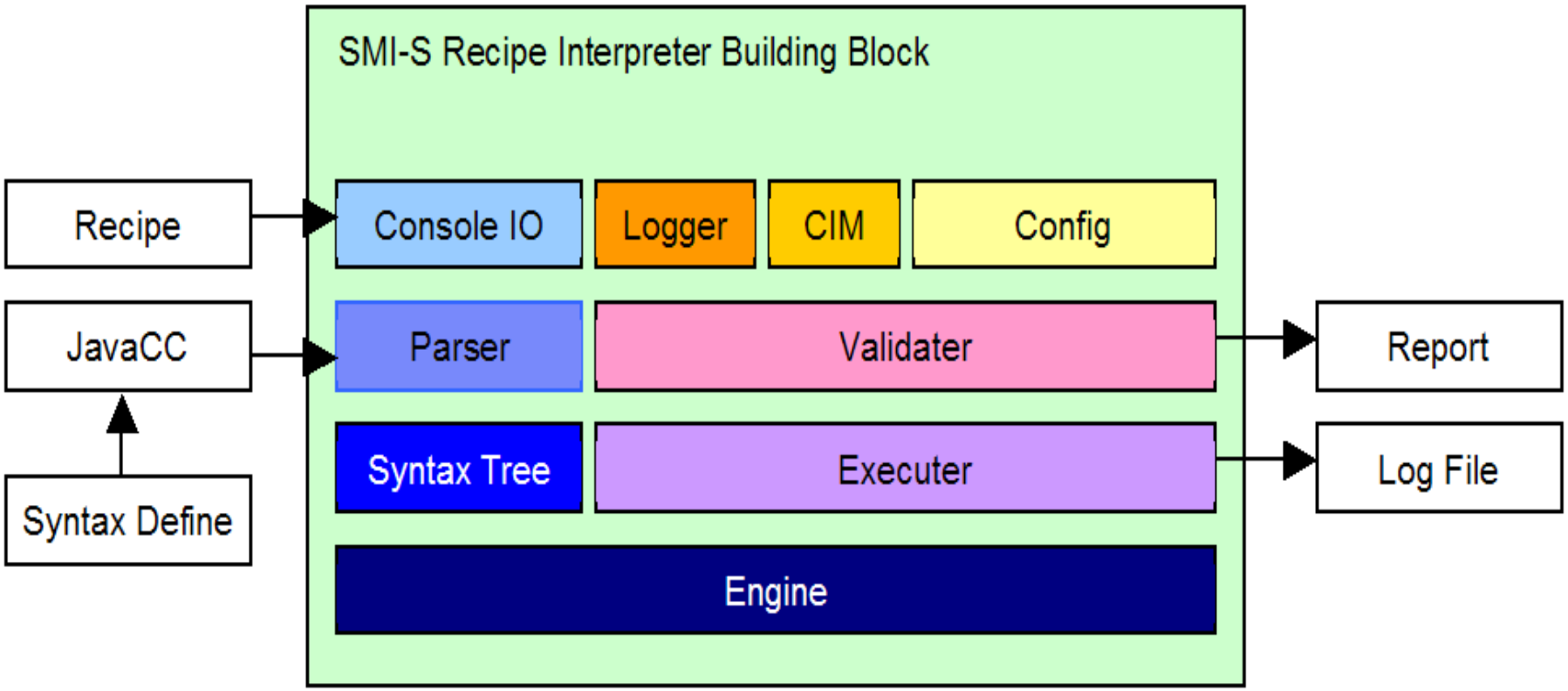
Demo



Agenda

- SMI Recipe Language (SMIRL)
- SMIRL Interpreter
- Demo
- **Interpreter Structure**
- Summary

Building Block



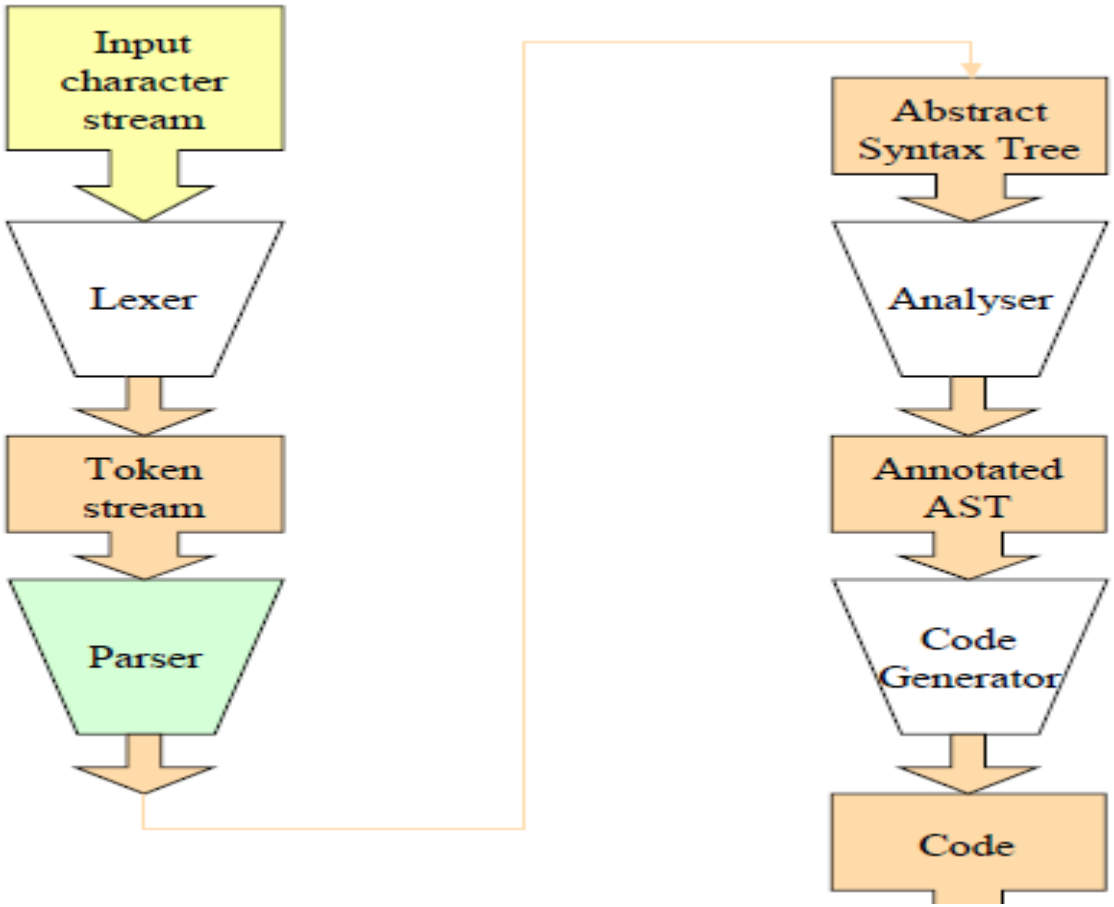
SMIRL Parser and JavaCC

JavaCC generate the parser class

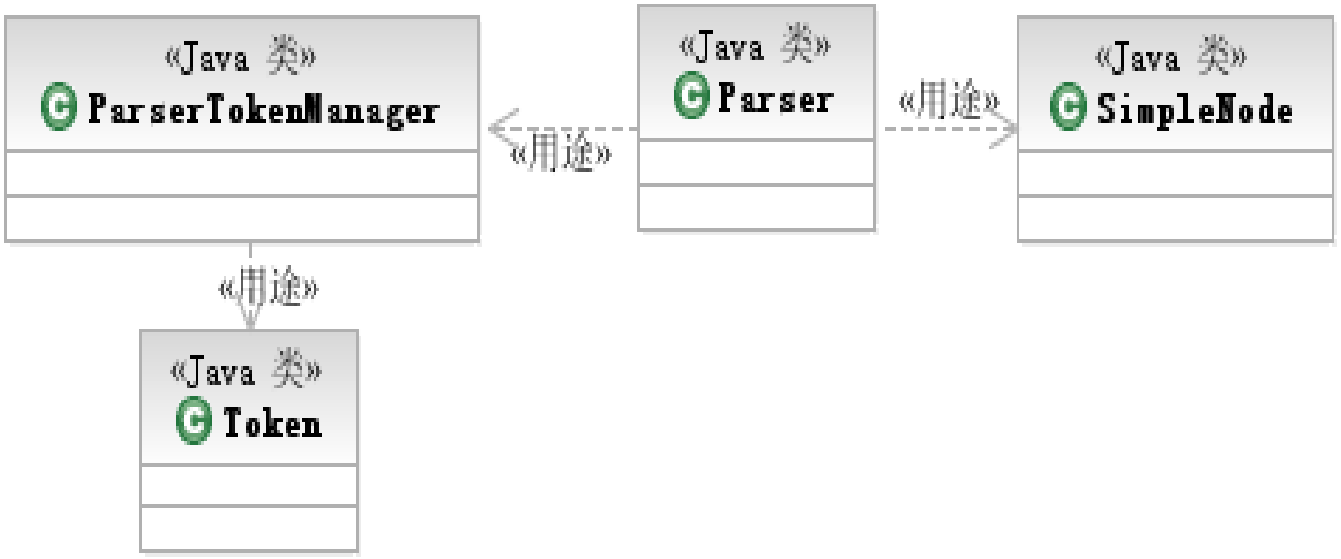
BNF production defined in .jj file as input

JavaCC translate .jj file into method

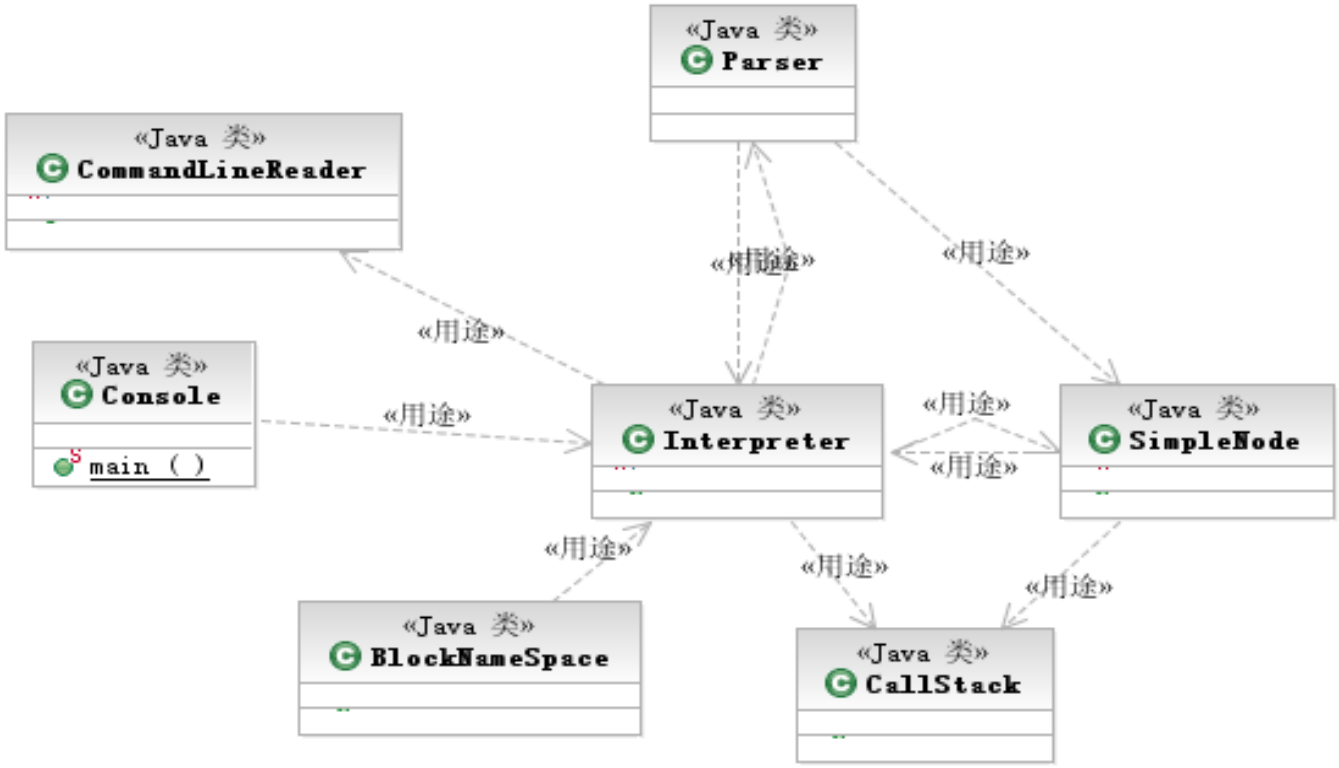
SMIRL Parser and JavaCC



SMIRL Parser and JavaCC



Interpreter



Agenda

- SMI Recipe Language (SMIRL)
- SMIRL Interpreter
- Demo
- Interpreter Structure
- **Summary**

Benefits

- Benefits
  - The tool implements an interpreter, which can read and execute Recipe syntax directly. People can execute the Recipe on specific target system, and do not have to understand the Recipe logic and translate that to other executable like JAVA.
  - The tool generates verification points according to the Recipe directly. User can control the validation point with configuration
  - The tool can generate execution log and error report to help user find the un-compliance positions.

Limitation

- In order to compress the document, some recipes are implied or assumed. This would include, for instance, that the set of available, interoperable properties are those explicitly defined by a particular profile or sub-profile.

- Indication still not supported by the tool

Thanks!