YangIDE Proposal

Name

YangIDE

Repo Name

yangide

Description

The YangIDE subproject will provide an Eclipse plugin that can be used to view and edit Yang model files. The following basic features are aimed to be supported:

File and Project Support

- Create Yang file
- Create Yang Maven project
- Create Yang project (project without Maven integration)

Code Editing

- While editing or viewing a Yang model file, symbols and keywords will be color-coded to provide visual differentiation.
- While entering new Yang keywords or user-defined symbols (module names, type names, et cetera), “smart completion” will be available, to both save typing time and remind users of the exact keywords and user-defined symbols.
- The editing view will immediately notify the user when the text they have entered is either syntactically or semantically invalid. Syntactic checking will be based on compliance to the Yang 1.0 specification. Semantic checking will report issues that would only be seen by a full Yang compilation, like mistyped module or type names.
- While viewing the reference to an imported module or type reference, a keyboard shortcut (or menu item) will change the editor view to display the defined module or type definition. Another keyboard shortcut will return the user to the original file. This view of defined modules or types will work whether the module is present in source form in the project itself, or provided by an external import source, like Maven dependencies, Github sources, or local project dependencies.
- The editor will provide “refactoring shortcuts”. This will include renaming symbols causing all the references to effect the same change, along with advanced refactoring features like selecting multiple leaf items and replacing it with the definition of a grouping and the reference to that grouping.
- Hovering the mouse over a reference to a symbol will display in a tooltip the description of that symbol.
- While viewing a large Yang model file, hierarchical constructs can be “folded” to provide a wider view of the entire file. A separate “Outline view” will provide a high-level list of the components in the file.
- To save time entering common constructs, users can define templates that are inserted by entering a single symbol along with a keyboard shortcut. The plugin will come with a full set of these, and the user can easily add to them.
- Preferences for the plugin can specify properties of the Yang code formatter, with options like presenting imports on a single line, or wrapping long description strings, along with formatting the entire file with reasonable indenting.

Diagram Model Editor

- In addition to the textual Yang model view, a graphical “UML-like” parallel editor will be available, allowing navigation and editing of Yang models in a “point-and-click” fashion. This will be integrated directly with the text editor view, so changes made in the text view will be immediately visible in the diagram view, and vice versa.

Export Features

- Export Yin (XML) file
- Export UML view of Yang model

Maven integration

- The editor will provide a wizard to use the “Maven Yang Plugin” to generate Java code associated with the Yang model.
- Import Maven project.
- Yang module dependencies will be resolved through Maven dependencies specified in project.

non-Maven integration

- Specify github (or more general) URLs to provide import sources
- Use Eclipse project dependencies for additional import sources

Scope

The YangIDE application will use the Yangtools ODL subproject for parsing and semantic validation of Yang models. At the present time, there are no obvious dependencies on the YangIDE project from other parts of OpenDaylight. There is no reason to disallow this reuse, but to be practical, anything that is part of YangIDE that is found to be reusable would be better exported to a shared component that the YangIDE project depends on.

Resources Committed (developers committed to working)

David Karr (dk068x@att.com)
Kevin D'Souza (kd6913@att.com)
Jonathan Pang (jp926h@att.com)

Initial Committers

David Karr (dk068x@att.com; gerrit: davidmichaelkarr) will be the initial committer.

Vendor Neutral

The initial codebase being used to seed the project will be completely open-source (Eclipse Public License), with no proprietary logos or artifacts.

Meets Board Policy (including IPR)

Yes