1. Document Specification

Version: 1.0
Date: Sep. 24th. 2011

2. Installation

ATS-IDE is an Eclipse plug-in. You need an Eclipse to start using it. We recommend Eclipse with version no earlier than 3.5. For a simple installation process, the current Eclipse (Eclipse Indigo 3.7) with C/C++ Development Tooling (CDT) is highly recommended. It can be downloaded from the following web address
http://www.eclipse.org/downloads/. (Please choose “Eclipse IDE for C/C++ Developers (includes Incubating components)”.)

The following steps show how to install ATS-IDE on Eclipse Indigo 3.7. To install ATS-IDE on your existing Eclipse may take some extra steps, which are also stated below.

Step 1. Click on Menu Help -> Install New Software, you will see the dialogue (See Figure. 1).

Step 2. Click Add. Input the address for the update site of ATS-IDE (See Figure. 2). Click OK.
Step 3.
If you are working on Eclipse Indigo 3.7, you can skip this step. Otherwise you need to add the update site for Xtext. Repeat Step 2, but input the address for Xtext update site instead. This address can be obtained from http://www.eclipse.org/Xtext/download/. The current one is http://download.eclipse.org/modeling/tmf/xtext/updates/composite/releases/. (See Figure. 3)

Step 4.
Choose to work with the newly added ATS-IDE update site and select the ATS IDE plug-in. Click Next (See Figure. 4)
Step 5.
Keep going to confirm license, indicate trust to the software and etc. Click Finish in the end and choose to restart Eclipse when being prompted.

3. Update

We will release newer version of ATS-IDE through the same update site. Please keep you ATS-IDE updated. You can follow the Step 4 - 5 of Installation. Eclipse will detect whether an older version of ATS-IDE has been installed and decide automatically whether to do the installation or update.

Another way is to click on “What is already installed?” in the dialogue of install (See Figure. 5) and choose ATS-IDE in the new dialogue (See Figure. 6) and click update.
Figure. 5
4. **Remove**

Similar as update, but click Uninstall after selecting ATS IDE (See Figure. 6).

5. **Install CDT (C/C++ Development Tooling)**

For just editing ATS code, any kind of project in Eclipse is sufficient. However, it’s more convenient to hold a bunch of ATS files along with corresponding Makefile in a C Project. Therefore we suggest that ATS users should install CDT. The installation process is similar to that of ATS-IDE. The update site is provided on the CDT website.

6. **How to use ATS-IDE**

Now we assume that we are working on the Eclipse with CDT installed.
1) **Create C project to hold ATS code**

Step 1.
Click Menu File -> New -> C Project, input the Project name, select Makefile project (Empty Project) and then click Finish (See Figure. 7).

![Figure. 7](image)

Step 2.
Right click on the project name, and then choose New -> File. Input the name for ATS files (See Figure. 8)
Step 3.
When prompted for Xtext nature, please choose Yes (See Figure. 9). This will only be prompted once per project.

Step 4.
Now you can try editing ATS file in Eclipse with the new ATS-IDE plug-in.
2) **Add ATS library**

If you want the cross-reference to ATS library files, you need to link them into the project.

Step 1.
Right click on the project name, select New -> Folder. In the New Folder Dialogue, click Advanced and then select “Link to alternate location” (See Figure. 10). Input the correct path for the directory “prelude” in the ATS installation directory.

Step 2.
Link a folder into project will import all the files in that folder into the current project as resources. This may slow down the build process of the project. For cross-reference to ATS library, we only need the sats file. Therefore we can create Resource Filter to filter out all the other files.

Step 3.
To create a Resource Filter, click on Resource Filters on the New Folder Dialogue (See Figure. 10). In the new dialogue for Edit Resource Filters, click Add (See Figure. 11). Then make sure you make the selection as shown by Figure. 12. Then click OK.
Figure. 10

Image of a dialog box for creating a new folder. The folder name is 'prelude'. The dialog box is showing the address '/home/atsuser/atshome/prelude' as the path.
3) **Makefile for ATS**

Create a Makefile in the current project. Figure. 13 is a very simple version for beginners of Makefile and ATS.

```
ATSCC=atscc
```

```
In this Makefile, atscc is called to compile the ATS code. Therefore the user has to make sure that ATS is installed and the executables of ATS including atscc as well as atsopt are visible to the Eclipse process.

**4) Create Targets for make**

Right click on the Makefile in the Project Explorer and then click Make Targets -> Create. Input the appropriate Target name (See Figure. 14). Then we
can double click on the target name in the Make Target View on the right side of the Eclipse window (See Figure. 15). The output information will be shown in the Console view at the bottom of the Eclipse window.

Figure. 14

Figure. 15
7. Functionalities of ATS-IDE

1) Syntax Highlighting

Users of ATS-IDE can configure various Token Styles for sats as well as dats file separately. The configuration can be done by clicking Menu Window -> Preferences and selecting AnairiatDats or AnairiatesSats on the new Preferences dialogue (See Figure. 16).

![Figure. 16](image)

2) Outline

While editing the ATS source file, the corresponding outline of the program is shown in the Outline view towards right in the Eclipse window. The Outline view can be moved around, docked, removed and added just as any other Eclipse view window. (For example, if you happen to close the Outline view, you can open it by clicking Menu Window -> Show View and then selecting Outline.). Double clicking on the node in the outline tree will highlight the corresponding code in the program (See Figure. 17). You can also expand the outline tree of the program to have a detailed knowledge of the underlying ATS syntax.
3) **Cross Reference**

When you hold the Ctrl and move the cursor over certain element of the ATS program you are editing, the underlining element would turn into a hyperlink. You can then left click on the hyperlink. The ATS-IDE will open the corresponding file if necessary, and then locate and highlight the corresponding element (which is to be referenced). We refer to this process as finding the reference of element or jumping from element to another.

**What Cross Reference can do**

ATS-IDE can only find reference for elements in the dynamics of ATS. It can jump from function implementation to its declaration, from value (including function, constructor and common value) to its definition and from the name of the loaded file (staload / dynload) to the actual file.

**What Cross Reference cannot do**

ATS-IDE cannot find reference for elements in the statics of ATS (e.g. it cannot jump to the definition of certain type from its name). It cannot find reference for elements used in pattern matching.

If the element to be referenced is in another file, then such file must be “staload”ed in the current file directly. For example, if the current file uses an element declared in another file “a.sats”, then the current file should contain ‘staload “a.sats”’. This restriction is much stronger than that imposed by ATS compiler. For the current file to be compiled by ATS compiler, it’s O.K. that “a.sats” is “staload”ed by another sats
file which is in the end “staload”ed in the current file. The exception of the rule is that for ATS-IDE to find cross reference in the library file, there is no need to “staload” default library sats file in the current file. This is consistent with the behavior of ATS compiler, e.g. we don’t need to “staload” the default library file “list0.sats” to use the constructor “list0_cons”.

Sometime cross-reference may not work properly when there exists syntactic error in the current source code. Try padding the rest part of source code you are working on, and then invoke cross reference.

4) **Content Assistant**

Normally the hot key for triggering content assistant is Ctrl+Space. But on certain machines it may be Alt+/.

At certain positions of the ATS program, type the prefix of certain word (or doesn’t type anything) and then press Ctrl+Space, ATS-IDE will prompt all the possible references which are visible to the current location and are of the same prefix as user has just typed (See Figure. 18).

![ATS-IDE screenshot](image)

**Figure. 18**

ATS-IDE may also prompt proposal for code template (e.g. the “if expression” in Figure. 19).
Once the user chooses it, the code template would be inserted into the current location (See Figure. 20).

Users can add and change template proposals in the preference page. To do this, click Menu Windows -> Preferences, and on the popped up windows select AniaristsDats -> Templates (See Figure. 21). A possible template configuration for “if expression” would be like Figure. 22.
Please refer to Eclipse online help as well as Xtext document for information about how to write such template. We will provide document about writing template for ATS and ship more useful template with ATS-IDE in the future ATS-IDE release.
5) **Toggle Comment**

Use cursor to select multiple lines and then press Ctrl+/ , ATS-IDE will toggle these lines into comments (See Figure. 23). Press Ctrl+/ again shall toggle comments back to original code.

![Figure. 23](image)

6) **Error Message**

Currently, ATS-IDE can report lexical and syntactic error inside ATS source code with red-underline. The syntax behind ATS-IDE is slightly different from the one recognized by ATS compiler. Therefore even if the ATS-IDE doesn’t report any error, the source file may still contain syntactic error. On the contrary, ATS-IDE may report error for certain ATS source code which is deemed valid by ATS compiler (we try to eliminate such cases as much as possible). Moreover, no semantic error is detected at all by the ATS-IDE. Hence user should not rely on ATS-IDE for the validation of their ATS source file. We suggest that when editing ATS source code, make sure ATS-IDE doesn’t report any lexical or syntactic error, try to eliminate errors of “Couldn’t resolve reference to refentity ‘xx’.” (by add “staload” into current source file) and then use ATS compiler to compile (or type check) the source file.

8. **To do list for ATS-IDE 2**

ATS-IDE is based on Xtext. The parsing of ATS source file as well as the processing of the corresponding syntax tree are handled inside the Java plugin. So far such processing does little semantic analysis. The design principle of ATS-IDE 2 is to make use of ATS compiler as much as possible to do such processing so that the ATS-IDE can have better analysis result.
1) **Semantic Highlighting**

Highlighting for keywords is still handled by Xtext instantly during user’s input. At certain point, use ATS compiler to scan the whole source file to get more precise information about the location and types of semantic tokens.

2) **Cross Reference**

ATS-IDE 2 should support cross reference for 1) static items (types, sorts and etc.); 2) elements constructing pattern match; 3) elements inside file which is indirectly “staload”ed into the current file; 4) elements with qualified name. In the current ATS-IDE, the name assigned to “staload”ed module as well as the prefix of a qualified name is neglected when resolving the cross reference. This may lead to incorrect cross reference when multiple files with elements of the same name are “staload”ed.

For overloaded symbol, the current ATS-IDE can only jump to the location where the symbol is introduced by “symintr”, not the real function. For template function, it can only jump to the declaration of the template, not the actual instantiation function. ATS-IDE 2 is expected to do these correctly.

3) **Validation (Type Checking)**

Use ATS compiler to type check the whole source file, and show corresponding error information (red underline and error message) in the editor.

4) **Formatting (Pretty Printing)**

Support the usage of Ctrl+Shift+F so that users can reformat certain part of the source code more easily.

5) **Content Assistant**

1) Ship with more template for content assistant. 2) Support dynamic template. For example, when prompting function names for user to select, also show the type of the function. Once user selects certain function, insert the template for making function call pertaining to this specific function.

6) **Quick Fixes**

For certain error, add suggestion for users to fix the problem.
7) **Type Inference for Selected Expression**

When user selects certain expression and hover the cursor over it, the editor prompts the type of the expression.