ASN.1 Support

Initial Abstract Syntax Notation 1 (ASN.1) support has been added to T3Q in version 2.1.0b4. It relies the compiler from [Eclipse TITAN](?https://projects.eclipse.org/projects/tools.titan), which needs to be installed separately. The path to the `compiler` executable needs to be specified in the corresponding configuration entry in the profile (**`titanCompilerPath`**) which is set to `compiler` by default, assuming that the compiler executable is included in the system path and readily executable from any location. If that is not the case, the correct path needs to be specified in the profile.

Stages

The use of ASN.1 definitions is a two-stage process:

1. Collect all ASN.1 files (file extension `.asn`) in the input path and convert them to a JSON Schema with the Eclipse TITAN compiler (option `--convert-asn1-to-schema`).
2. Generate corresponding TTCN-3 modules for each ASN.1 file from the JSON Schema produced by the Eclipse TITAN compiler (option `--convert-schemas`).

Each stage can be executed independently with the corresponding option. If both options are provided, they are executed in the order listed above. The Eclipse TITAN compiler is only required for the first stage.

The first stage produces an `asn.json` file in the current working folder where T3Q is executed, which contains the definitions from all the input ASN.1 files. The second stage uses the `asn.json` file and generate one TTCN-3 file in the current working folder for each ASN.1 file with the following convention `asn.json.<MODULE_NAME>.ttcn`, where `MODULE_NAME` is the generated TTCN-3 name corresponding to the ASN.1 file. If both stages are executed together, T3Q takes care of passing on the generated files from one stage to the next. If only the second stage is executed, then the `asn.json` file needs to be provided as part of the input paths for T3Q. The generated files will be overwritten between runs.

If the ASN.1 files do not change or if Eclipse TITAN is not available on the used platform / computer, the generated TTCN-3 files can be used directly by including them into the input path. For example, running T3Q with the first stage on a folder that contains `ModuleA.asn` and `ModuleB.asn` (even deeply nested) produces an `asn.json` file in the current working folder which contains all the definitions from both `ModuleA.asn` and `ModuleB.asn` as a JSON Schema. e.g.:

```
java -Xmx3g -Xss512m -jar t3q.jar --convert-asn1-to-schema --config config/t3q-tf160-next.xml
```

If only the first stage is selected, only the `asn.json` file is generated and the corresponding references cannot be resolved yet. Therefore, it makes sense to run the first stage with only the ASN.1 files as input. It is best to add all

the necessary ASN.1 files as input rather than processing individual files one at a time, as unresolved dependencies between ASN.1 files would result in failure to generate the `asn.json` file, e.g:

```
java -Xmx3g -Xss512m -jar t3q.jar --convert-asn1-to-schema --config config/t3q-tf160-next.xml
"iwd-TTCN3-B2024-06_D25wk24/NR5GC_IWD_25wk24/Common/SuppServices/Common_Definitions_Arguments.
"iwd-TTCN3-B2024-06_D25wk24/NR5GC_IWD_25wk24/Common/SuppServices/LCS_Definitions_Arguments.asn
"iwd-TTCN3-B2024-06_D25wk24/NR5GC_IWD_25wk24/Common/NR_Defs/NR_Sidelink_Preconf.asn" \
"iwd-TTCN3-B2024-06_D25wk24/NR5GC_IWD_25wk24/Common/NR_Defs/NR_PC5_RRC_Definitions.asn" \
"iwd-TTCN3-B2024-06_D25wk24/NR5GC_IWD_25wk24/Common/NR_Defs/NR_RRC_ASN1_Definitions.asn"
```

Given that `LCS_Definitions_Arguments` depends on `Common_Definitions_Arguments`, leaving `Common_Definitions_Arguments` out will result in a failure to produce an `asn.json` file.

Running T3Q with the second stage, including `asn.json` as input produces `asn.json.ModuleA.ttcn` and `asn.json.ModuleB.ttcn` files in the current working folder, which are then added to the input paths and considered during the evaluation of the other TTCN-3 files, e.g. using the `asn.json` file generated above:

```
java -Xmx3g -Xss512m -jar t3q.jar --convert-schemas --config config/t3q-tf160-next.xml --profi
```

produces the corresponding TTCN-3 files `asn.json.Common_Definitions_Arguments.ttcn`, `asn.json.NR_Sidelink_Preconf.ttcn`, `asn.json.LCS_Definitions_Arguments.ttcn`, `asn.json.NR_PC5_RRC_Definitions.ttcn`, and `asn.json.NR_RRC_ASN1_Definitions.ttcn`. These can then be used in a subsequent runs by adding them to the input paths without the need to convert the ASN.1 / JSON Schema again. These can also be shared with other users that do not have Eclipse TITAN available. Running T3Q with the `asn.json` file in addition to the TTCN-3 files referencing the ASN.1 definitions as input with integrate the generated files and consider them during the evaluation of the non-generated TTCN-3 files.

The generated files should be excluded from the evaluation otherwise there may be warnings related to them which are not really relevant as the files are generated from ASN.1 and it might also take a longer to process them. The configuration profiles should therefore include a corresponding pattern in **`ignoredResourceRegExp`**, e.g. `(.*asn[.]json[.].*ttcn)`.

Remote usage for Stage 1

If Eclipse TITAN is not available for the first stage (e.g. due to compatibility, platform, or other concerns), T3Q has the option `--serve-asn1-compiler` to provide the first stage remotely, e.g. on another computer on online server which can run Eclipse TITAN and is reachable from the local computer. Two steps are necessary:

1. Run T3Q on the remote computer that has Eclipse TITAN installed with the option `--serve-asn1-compiler` which will expose an HTTP endpoint `/compile_asn` at port `3005`, which accepts `POST` requests carrying the ASN.1 files and provides a response carrying the resulting `asn.json`

file after running the Eclipse TITAN compiler.
2. Update the **`titanCompilerPath`** setting in the configuration profile to point to the endpoint on the remote computer `http://<IP or DOMAIN>:3005/compile_asn` and run T3Q on the local computer that does not have Eclipse TITAN available with the `--convert-asn1-to-schema` option (or also adding the `--convert-schemas` option for an integrated run).

Instead of running Eclipse TITAN locally, T3Q will connect to the remote computer and send the ASN.1 files included in the input paths. It can then proceed with the received `asn.json` file as if it were generated locally.

Limitations

There are some restrictions due to upstream issues with Eclipse TITAN (for example parameterised types), which affect the generated schema and TTCN-3 modules. These will be investigated and resolved in future releases.

The remote functionality is experimental and not yet ready for public deployment. It is intended to bridge circumstances where Eclipse TITAN is not available locally. If needed, authentication and further features can be added to make it more robust and publicly deployable.

As ASN.1 files often include thousands of definitions, resolving and processing these definitions, especially in a single file can impact the overall duration of processing. If quick feedback is desired or the duration is a concern, preliminary check without ASN.1 support is sufficient for most quality checks as they do not rely on ASN.1 definitions. Some checks such as `checkNoUninitializedFieldsInTemplates` benefit from the ASN.1 definitions as otherwise templates for types defined in ASN.1 cannot be checked.