TTF 013: TDL and TOP Enhancement

Status Update

Document History

- 2021-09-24: Document submitted for MTS#84
- 2021-05-10: Document submitted for MTS#83
- 2021-03-09: Document submitted for TTF 013 Kick-Off Meeting

From the Terms of Reference…

TTF 013: Objectives (enhancement)

- Support for testing RESTful API services specified with OpenAPI
  - data definition refinements
  - workflow, TD generator, test execution engine
- Textual syntax standardisation
  - clean-up and refinement of Annex B (also of Part 4)
  - integration of Part 1 and Part 4 syntax? (by extension also Part 7?)
- Methodology for deriving TDs from TOs/TPs
- Initial work on testing of AI systems (requirements)

TTF 013: Objectives (maintenance)

- Adaptation and extension of TDL addressing CRs
  - new features (parametrisable TOs, reusable events, OpenAPI-related)
  - updates to relevant parts, mapping new features to TTCN-3
- Update of TOP tools
  - resolve CRs, catch up with Eclipse updates, adaptations to TDL
- Update of technical report
  - validation, user guidelines

TTF 013: Deliverables

Deliv.	Work Item code Standard number	Working title Scope
D1*	RES/ES 203 119-1 V1.6.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 1: Abstract Syntax and Associated Semantics
D2*	RES/ES 203 119-2 V1.5.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 2: Graphical Syntax
D3*	RES/ES 203 119-3 V1.5.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 3: Exchange Format
D4*	RES/ES 203 119-4 V1.5.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 4: Structured Test Objective Specification (Extension)
D5*	RES/ES 203 119-6 V1.3.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 6: Mapping to TTCN-3
D6*	RES/ES 203 119-7 V1.3.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 7: Extended Test Configurations
D7	DES/ES 203 119-8 V1.1.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 8: Textual Syntax
D8	RTS/TR 103 119 V1.3.1	Methods for Testing and Specification (MTS); The Test Description Language (TDL); Reference Implementation and User Guidelines

* Work items of the TDL standard series which are not affected by CRs will not be updated.

Today

- Work plan
- Tasks and status
- Open questions

Planing Summary

Work Plan

- Timescale: Mar, 2021 — Apr, 2022
  - work remotely, weekly online meetings, meet in person only if necessary
    - given current circumstances no in-person meetings before ~~summer/~~autumn 2021
    - coordinated remote sessions scheduled as needed, based on availability
- Working sessions
  - WK15: Apr 12–16 Focus Week @ Home (5 days)
  - WK36–WK37: Sep 8–14 Focus Week @ Home (4/5 days)
  - November: Working session / Focus Week (exact dates TBD)

Milestone Definitions

Task / Mil.	Destination	Description	Cut-off Date
MA	ETSI	Early drafts of D1, D2, D4, D6 available First progress report to be approved by TC MTS	15/05/2021
MB	ETSI	Stable drafts of D1 – D7 and early draft of D8 available Second progress report be approved by TC MTS	15/09/2021
MC	ETSI	Final drafts of D1 – D8 submitted to TC MTS	21/12/2021
MD	ETSI	Deliverables D1-D8 approved by TC MTS Final report to be approved by TC MTS	31/01/2022
ME	ETSI	Deliverables published, TTF closed	30/04/2022

Milestones (Current Planning)

N	Task / Milestone / Deliverable	ToR Targets	PM / Current Targets
	Start of work	01-Mar-2021	08-Mar-2021
T0	Project Management	01-Mar-2021–30-Apr-2022	08-Mar-2021–30-Apr-2022
T1	TDL Evolution	01-Mar-2021–31-Dec-2021	08-Mar-2021–31-Dec-2021
T2	TOP for RESTful API Services	15-May-2021–31-Dec-2021	15-May-2021–31-Dec-2021
T3	TDL Methodology	15-May-2021–31-Dec-2021	15-May-2021–31-Dec-2021
MA	Early drafts/report (MTS#83)	15-May-2021	15-May-2021
MB	Stable drafts/report (MTS#84)	15-Sep-2021	15-Sep-2021
MC	Final drafts (before MTS#85)	20-Dec-2021	20-Dec-2021
MD	Final report (MTS#85)	31-Jan-2022	31-Jan-2022
ME	Deliverables published	30-Apr-2022	30-Apr-2022

* Preliminary work on T2 and T3 started informally before MA as it informs work on T1 as well.

Overall Timeline (Current)

Task / Mil.	Description	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J
T0	Project Management																
T1	TDL Evolution																
T2	TOP for RESTful API Services																
T3	TDL Methodology																
MA	Early drafts/report (MTS#83)																
MB	Stable drafts/report (MTS#84)																
MC	Final drafts (before MTS#85)																
MD	Final report (MTS#85)																
ME	Deliverables published																

Tasks

Task 0: Project Management

- Two TTF working sessions (1x per milestone)
  - Additional working sessions to be scheduled as needed.
- Progress reports for Milestones B (submitted for MTS#84)
- Communications
  - Open meetings: 10-Mar-2021, 19-May-2021, 7-Jul-2021, 14-Jul-2021
  - Board SOOS presentation: 20-May-2021
  - Presentation at the UCAAT 2021 accepted, prepared, recorded
  - TTF Website: https://portal.etsi.org/STF/STFs/STF-HomePages/T013

Board SOOS Report on TOP

- Report to Board SOOS: 20-May-2021
  - Report on the TOP project, how it is managed, relation to specifications
  - Well received, lively discussion afterwards
  - Outcome may be relevant to TOP in the future
  - Invitation to continue to follow activities -> anyone available?
- @Ultan: Any updates from subsequent SOOS meetings?

Task 1: TDL Evolution

- Standardised textual syntax for TO and TD, indentation-based
- Parameterizable test objectives
- Re-usable events in TDL-TO
- New language features for supporting RESTful API services testing through their OpenAPI interface specification
- Resolving open CRs
- Design of new features according to the submitted CRs

Status: TDL Evolution

- Textual syntax draft proposal (Part 8)
  - both brace-based (default) and indentation-based delimiters for blocks
  - alternative notation for select constructs (Message, ProcedureCall)
  - natural vs programming language feel (currently still a bit of a mix)
    - natural language for TO, programming language for TDs?
  - completeness: mostly complete, with some restrictions (e.g. DataUse)
  - case-insensitive keywords

Status: TDL Evolution

```
Package returns tdl::Package:
    AnnotationFragment
    'Package' name=Identifier
    BEGIN
        (^import+=ElementImport)*
        (packagedElement+=PackageableElement)*
        (nestedPackage+=Package)*
    END
    WithCommentFragment?
;
```

```
Package CombinedBehaviours {
    Import all from DataUse
    Import all from DataDefinition
    Import all from DataUse.DynamicDataUse
    Import all from TestConfiguration

    TestDescription singleCombinedExample uses base {
        //…
    }
}
```

```
Package CombinedBehaviours
    Import all from DataUse
    Import all from DataDefinition
    Import all from DataUse.DynamicDataUse
    Import all from TestConfiguration
    TestDescription singleCombinedExample uses base
```

```
BEGIN : ‘{‘
END : ‘}’
```

```
BEGIN : <INDENT+>
END : <INDENT->
```

Status: TDL Evolution

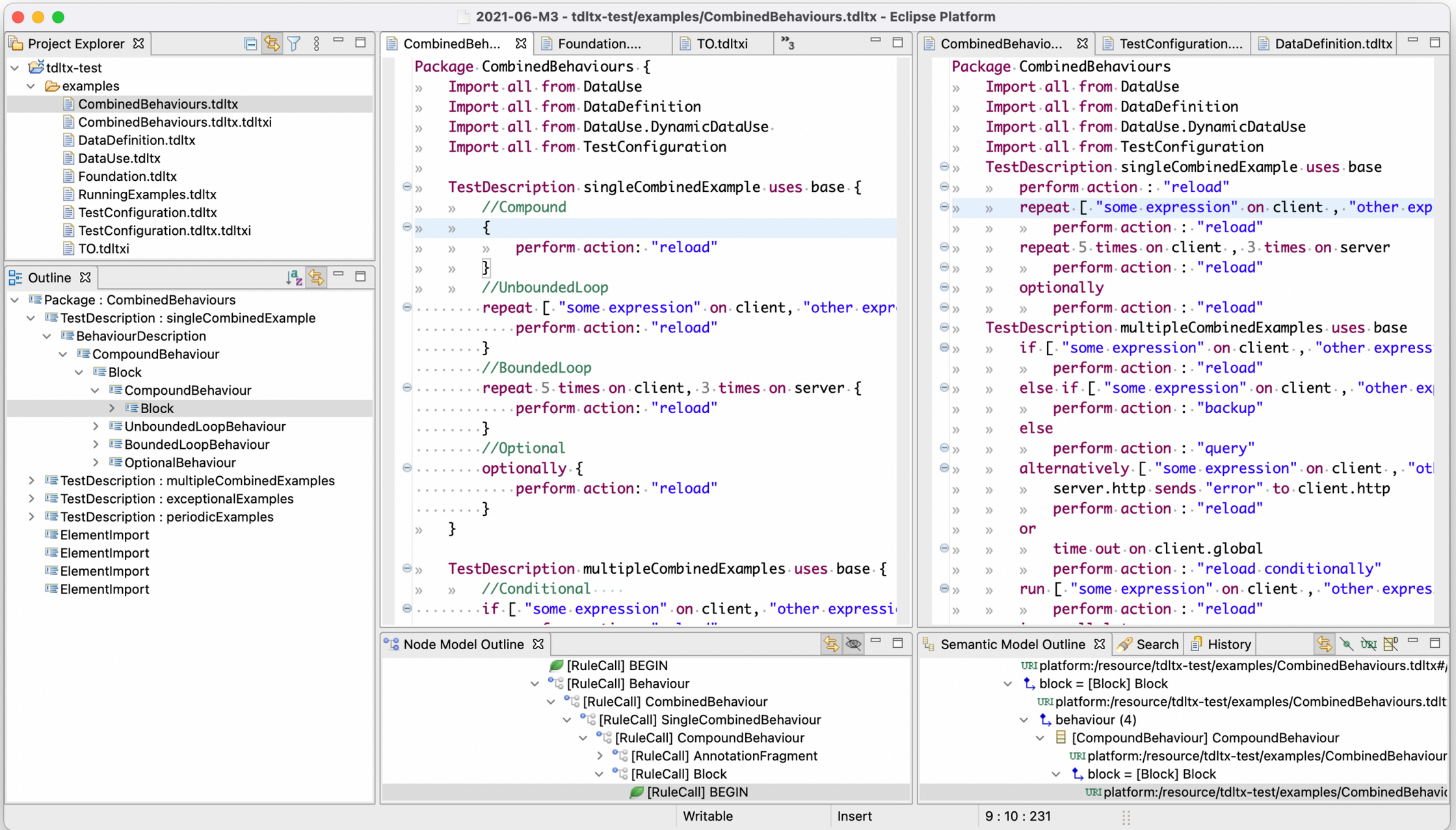
```
Package TestConfiguration {
    Import all from DataDefinition
    Import all from Foundation
    Message Gate HTTP accepts Post, Posts
    Message Gate HTTPS extends HTTP accepts Binary
    Procedure Gate RPC accepts publish
    Component Node {
        timer global
        variable Binary authToken
        variable Boolean authenticated
        variable Integer lastPostId
        gate HTTP http
        gate RPC rpc
    }
    Component SecureNode extends Node {
        gate HTTPS https
    }
    Configuration base {
        create SUT server of type Node
        create Tester client of type Node
        connect client.http to server.http
        connect cRPC=client.rpc to sRPC=server.rpc
    }
}
```

```
BEGIN : ‘{‘
END : ‘}’
```

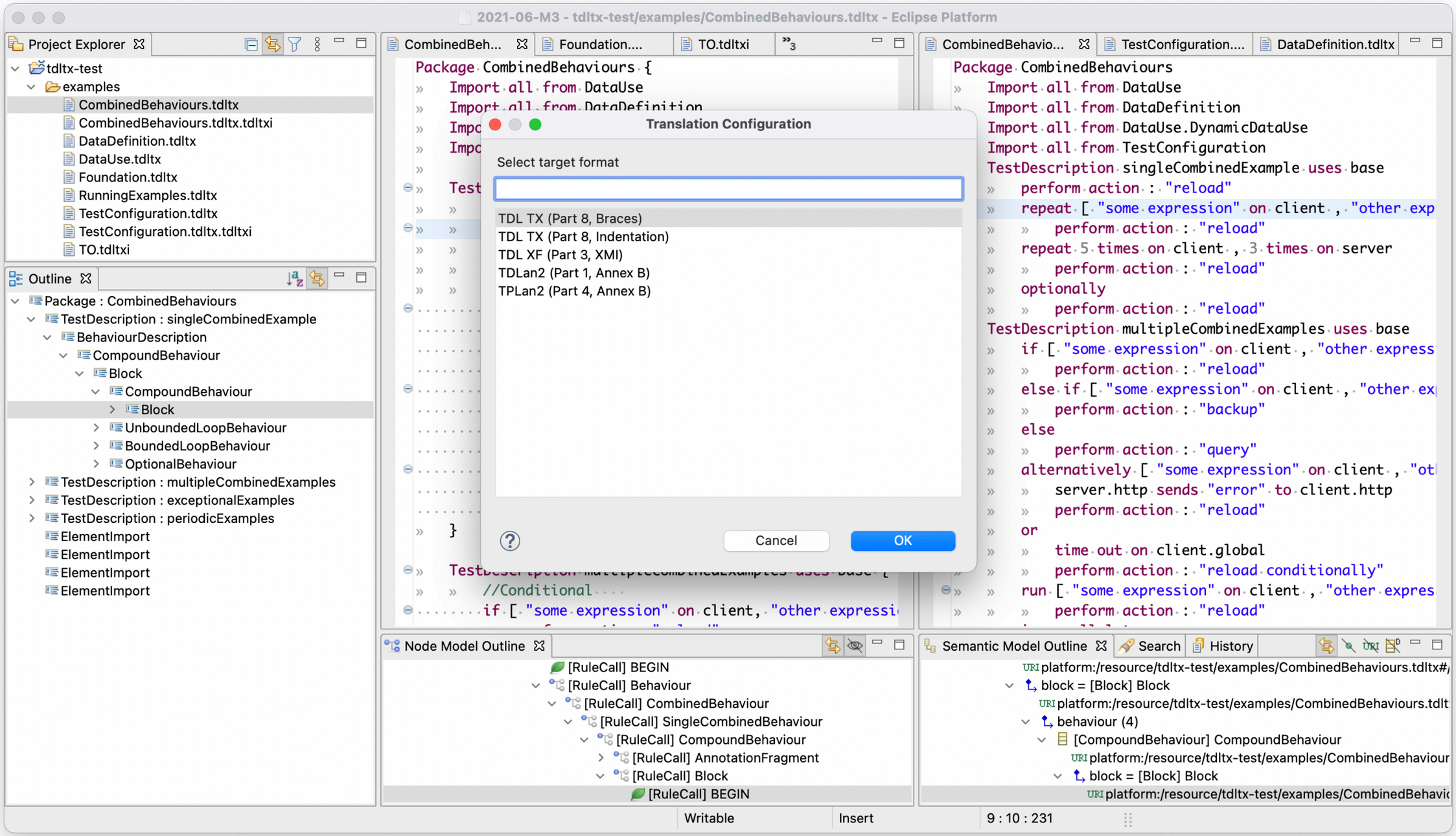
```
Package TestConfiguration
    Import all from DataDefinition
    Import all from Foundation
    Message Gate HTTP accepts Post , Posts
    Message Gate HTTPS extends HTTP accepts Binary
    Procedure Gate RPC accepts publish
    Component Node
        timer global
        variable Binary authToken
        variable Boolean authenticated
        variable Integer lastPostId
        gate HTTP http
        gate RPC rpc
Component SecureNode extends Node
        gate HTTPS https
Configuration base
        create SUT server of type Node
        create Tester client of type Node
        connect client.http to server.http
        connect cRPC = client.rpc to sRPC = server.rpc
```

```
BEGIN : <INDENT+>
END : <INDENT->
```

Status: TDL Evolution



Status: TDL Evolution



Status: TDL Evolution

```
//default
 server.http sends ? of type RestrictedListOfInt to client.http

//default + variable assignment
 server.http sends ? of type RestrictedListOfInt to client.http
     assigned to authToken

//alternative: swap source and target
 client.http receives ? of type RestrictedListOfInt from server.http

//alternative: swap source and target and assign to variable
 authToken = client.http receives ? of type RestrictedListOfInt from server.http
```

Status: TDL Evolution

- Other matters for discussion
  - Change title of TR 103 119? (still “Reference Implementation”)
  - Status of Annex B in Parts 1 and 4 (“unofficial textual syntax”)
    - Update, add note that it is superseded by Part 8, may be deprecated in the future
- Clause renumbering to bring related parts together
- Validation operation / alternatives for data -> discuss during WG meeting
  - Different scenarios for the specification of data alternatives
  - Introduce dedicated constructs vs guidelines for existing features

Task 2: TOP Maintenance

- Implement the elaborated workflow for RESTful API services testing from Task 3

- Provision of a TDL-TD code generator and execution engine to support the execution of RESTful API services tests

- TOP maintenance in accordance to updates of its base software (Eclipse)

- Resolving open CRs and alignment of TOP according to TDL changes performed in Task 1

Status: TOP Maintenance

- Prototypical implementations for Part 8
  - conversion between different notations for validation
- Prototypes for transformation of OpenAPI and ASN.1 data definitions
- Prototypes for interpreter/code generator for tests for RESTful APIs
- Prototype for transforming TOs/TPs to TDs
  - data, configuration, behaviour stubs

Status: Methodology

```
NodeDescriptor ::= SEQUENCE
{
    aNodeName NodeName,  -- Node name
    aShortName UUID,  -- Short node name
    aNode CHOICE
    {
        aLink SEQUENCE
        {
            aLinkedFileIdentity NodeIdentity,  -- Identity of the linked SSP file
            aLinkedFileSize FileSize  -- Size of the linked SSP file
        },
        aFile SEQUENCE
        {
            aFileSize FileSize  -- Size of the SSP file
        },
        aDirectory SEQUENCE
        {
        }
    },
    aMetaData SEQUENCE OF MetaDatum OPTIONAL,  -- Optional meta data
    aACL SET OF AccessControl OPTIONAL  -- Access Control List attribute
}


/* Node identity */
NodeName ::= UTF8String (SIZE(1..16))  -- node name encoded in UTF-8
NodeReference ::= SEQUENCE (SIZE(1..6)) OF NodeName  -- pathname and node name

NodeIdentity ::= CHOICE
{
    aShortName UUID,  -- UUID of file reference using absolute pathname
    aNodeReference NodeReference  -- Node reference
}
```

ASN.1 Data Definitions

```
Type NodeDescriptor (
        aNodeName of type NodeName ,
        aShortName of type UUID ,
        aNode of type NodeDescriptor___aNode ,
        optional aMetaData of type NodeDescriptor___aMetaData ,
        optional aACL of type NodeDescriptor___aACL
) ;
Type NodeName with {
        UTF8String ;
} ;
Collection NodeReference of type NodeName ;
Type NodeIdentity (
        aShortName of type UUID ,
        aNodeReference of type NodeReference
) with {
        CHOICE ;
} ;

Type NodeDescriptor___aNode (
        aLink of type NodeDescriptor___aNode___aLink ,
        aFile of type NodeDescriptor___aNode___aFile ,
        aDirectory of type NodeDescriptor___aNode___aDirectory
) with {
        CHOICE ;
} ;
Type NodeDescriptor___aNode___aLink (
        aLinkedFileIdentity of type NodeIdentity ,
        aLinkedFileSize of type FileSize
) ;
Type NodeDescriptor___aNode___aFile (
        aFileSize of type FileSize
) ;
Type NodeDescriptor___aNode___aDirectory ( ) ;

Collection NodeDescriptor___aMetaData of type MetaDatum ;
Collection NodeDescriptor___aACL of type AccessControl ;
```

TDL Data Definitions (imported)

Status: Methodology

TO (inline data)

```
the IUT entity receives a SUBSCRIBE message containing
  payload containing
     filterLength corresponding to TOPIC_FILTER_LEN_SEC_CVE_01,
     topic_filter corresponding to TOPIC_FILTER_LEN_SEC_CVE_01;;
from the ATTACKER_CLIENT entity

Data {
  UTF8String TOPIC_FILTER_SEC_CVE_01;  // topic filter used in TP_MQTT_BROKER_SEC_CVE_001
  Int16 TOPIC_FILTER_LEN_SEC_CVE_01;   // corresponds to lengthof(TOPIC_FILTER_SEC_CVE_01) + 1
}
```

TDL (structured data)

```
Type SUBSCRIBE_message (
  payload of type SUBSCRIBE_message_payload
) ;
Type SUBSCRIBE_message_payload (
  filterLength of type Int16 ,
  topic_filter of type UTF8String
) ;
```

Task 3: Methodology

- Elaborate a defined way to derive TDs from TOs and provide guidelines for a semi-automatic workflow

- Elaborate a workflow to specify TOs/TDs for RESTful API services starting from an OpenAPI specification in accordance with EG 203 647

- Demonstrate and describe the application of TOP for RESTful API Services testing

- Update online documentation, wiki, examples

- Solicitation of requirements for describing tests of AI systems and ML models

Status: Methodology

- Working proposal for the derivation of TDs from TOs
  - alternative syntax TP-like syntax for TDs pending (best effort)
- Working proposal for the derivation of TDL data from ASN.1
  - similar description of the derivation of TDL data from OpenAPI pending

Open Questions

Informative vs Normative Work

- Some topics raise questions, may need a new home
  - OpenAPI / ASN.1 use and guidelines
  - TO to TD derivation
  - Use of TDs as containers for TPs
    - scope and limitations (constructs, semantics, etc.)
- Informative as a start?
  - Otherwise new WIs?
- New contact point at CTI (due to Michele’s departure)

Concrete Syntax

- Unified syntax and file types vs separate file types
  - currently separate file types -> separate editors, compatible references
  - Part 8 can unify the syntax -> implies support for Part 4 -> not at present!
- Braces vs indentation
  - need to better understand tradeoffs
  - supporting both possible -> current working proposal
    - can be considerable overhead both for specification and for implementation
    - also confusing for users?

Future

- New TTF ToR submitted, focusing on
  - Web platform for TDL and integration with NWM
  - TOP project refinements (execution architecture, code generation, UX)
    - Potential normative work: TRI, transformation rules (currently informative)
- TTF Roadmap needs updating

Any other business?