Towards a TTCN-3 Test Framework for OCCI-Based Cloud Ecosystems

1st User Conference on Advanced Automated Testing

UCAAT

Paris . 23 October 2013

Yongzheng Liang
University Stuttgart
liang@rus.uni-stuttgart.de

Overview

- Motivating assumptions
- OCCI
- TTCN-3
- Towards TTCN-3 / OCCI Ongoing/future work
- Acknowledgements
- Conclusion

Motivating Assumptions

- Cloud systems will successfully evolve beyond “simple” IaaS compute-network-storage scenarios
- OCCI is a candidate for related management tasks of IaaS, PaaS … XaaS
- There will be strong requirements concerning “assurance of ~ ” – “conformance”, “governance”, “performance”, “interoperability” etc.
- It will pay off to invest in related testing technologies
- TTCN-3 is a testing technology candidate in the envisioned context

OGF's Open Cloud Computing Interface - What is OCCI?

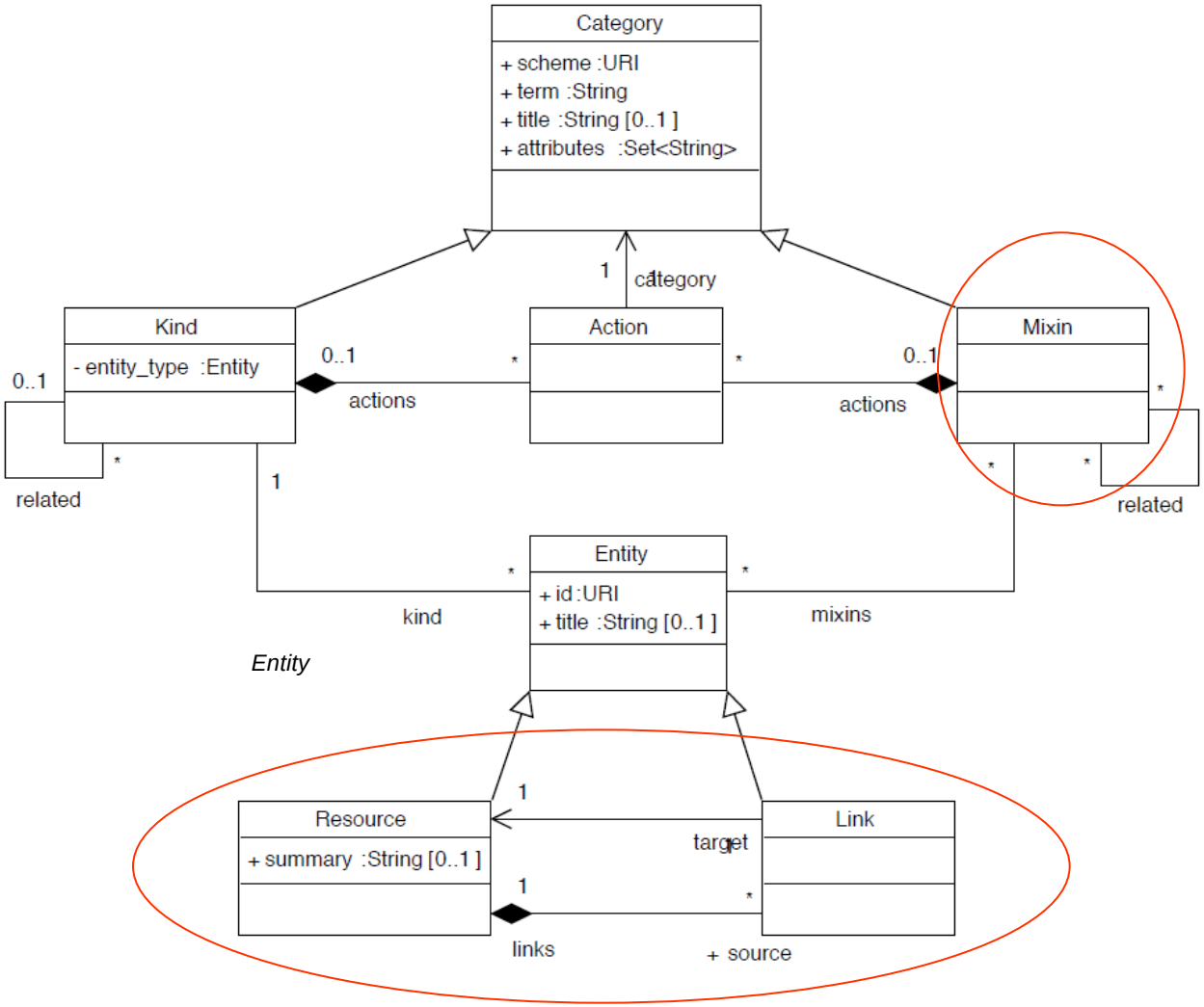
OCCI – the Open Cloud Computing Interface - is a

- RESTful Protocol and API for all kinds of management tasks.
- originally … management API for IaaS …
- development of interoperable tools for common tasks including deployment, autonomic scaling
- … high degree of extensibility.
- current release suitable to serve in addition to IaaS, PaaS and SaaS.

OCCI – present state

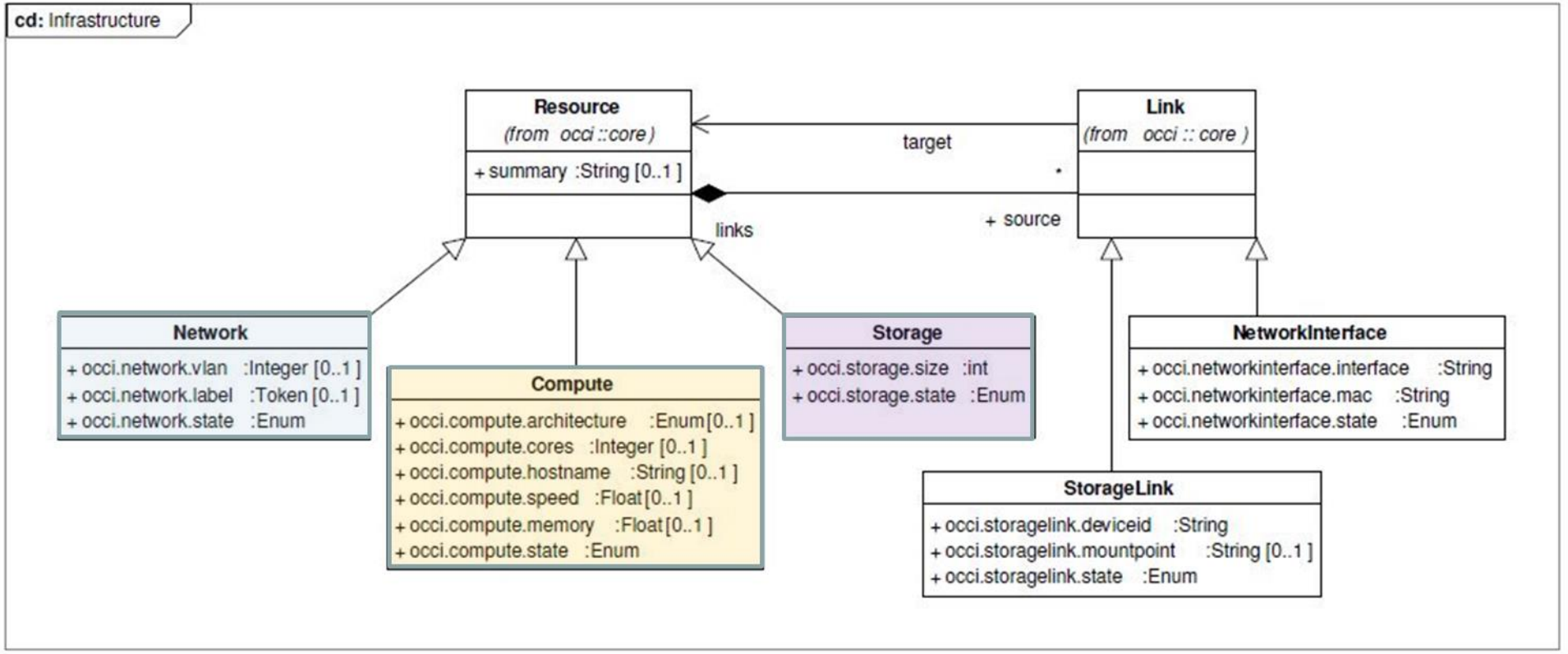
- OGF OCCI WG: 3 specifications:
  Core, Infrastructure, HTTP Rendering
  - Ongoing work on JSON Rendering, Monitoring, SLA, …
- OCCI server implementations used by OpenStack, FI-WARE, OpenNebula, …
- Outside the OCCI WG: Work on further extensions
- ‘Market’ success: 2nd after AWS

Overview of the OCCI Core Model [1)]

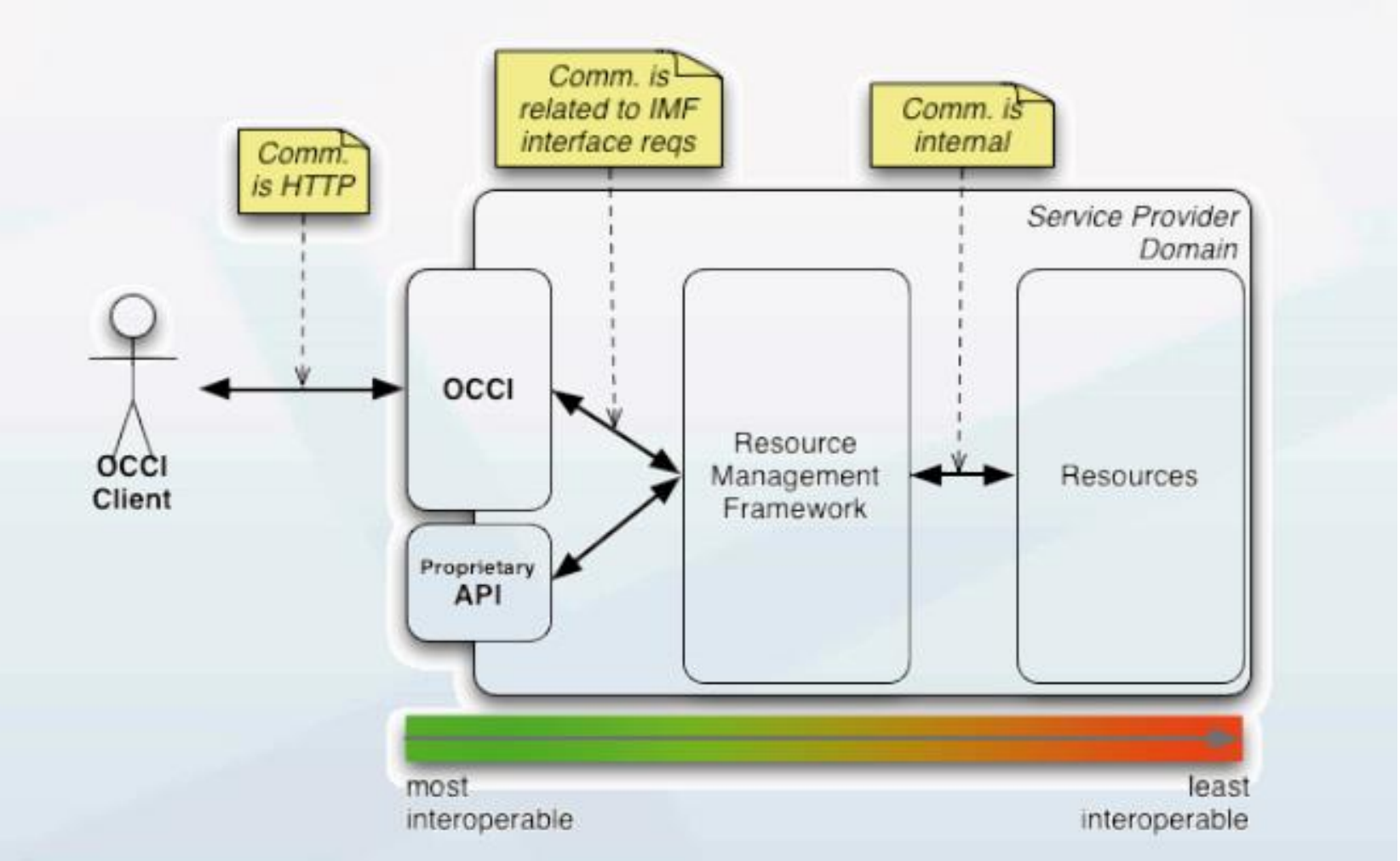


1): OCCI-WG: GFD-P-R.183; Figure 2 . April/June 2011

Overview of the OCCI Infrastructure Model

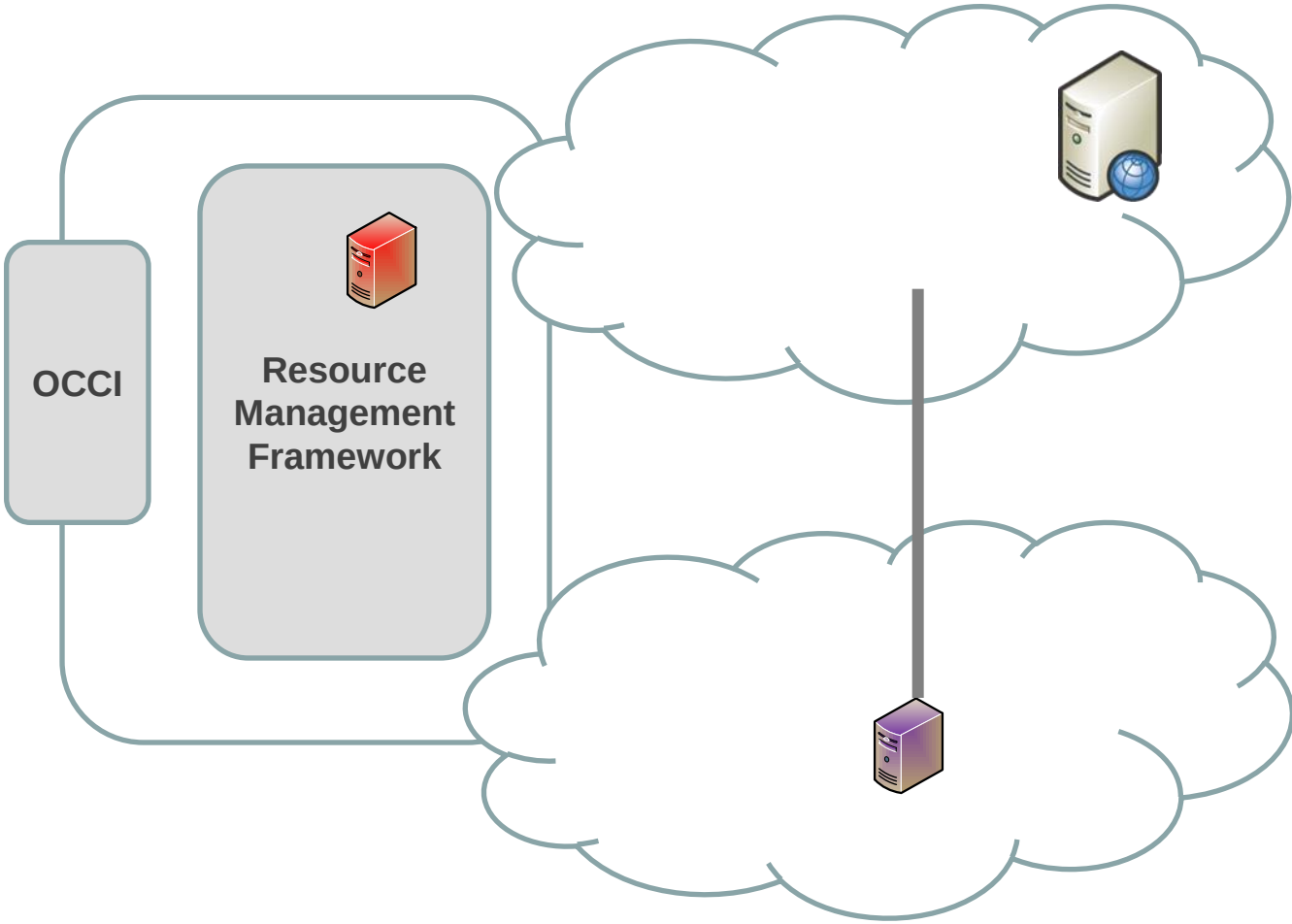


OCCI – Positioning -1(2)



OCCI – Positioning 2 (2)

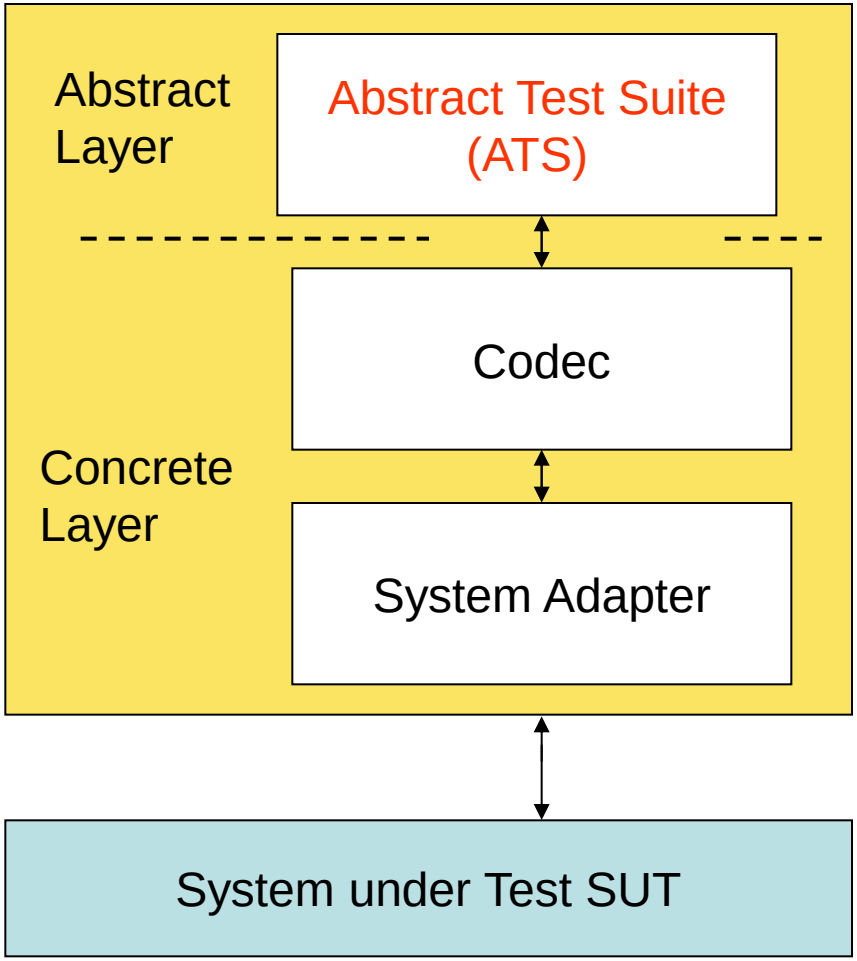
```
curl – X DELETE
     -H “Category: compute; …”
curl – X POST
     -H “Category: network; …”
curl – X POST
     -H “Category: compute; …”
     -H “X-OCCI-Attribute: …”
```



TTCN-3 - What is TTCN-3?

- TTCN-3 Testing and Test Control Notation Version 3 is a successful Test Specification Language standardized by ETSI
- Initial target: Telecom protocol conformance testing – e.g. IPv6, SIP, LTE
- Today: new technical domains – Web, embedded, real-time, … new sectors - Health, Automotive (Autosar), …
- Main characteristics:
  - Multi-Separation of Concerns:
    - Abstract but executable Test Specification Layer ("model-based testing")
    - Concrete Codec and Test-Adaptation Layers
  - Validation:
    - Template matching mechanism to validate output from SUT

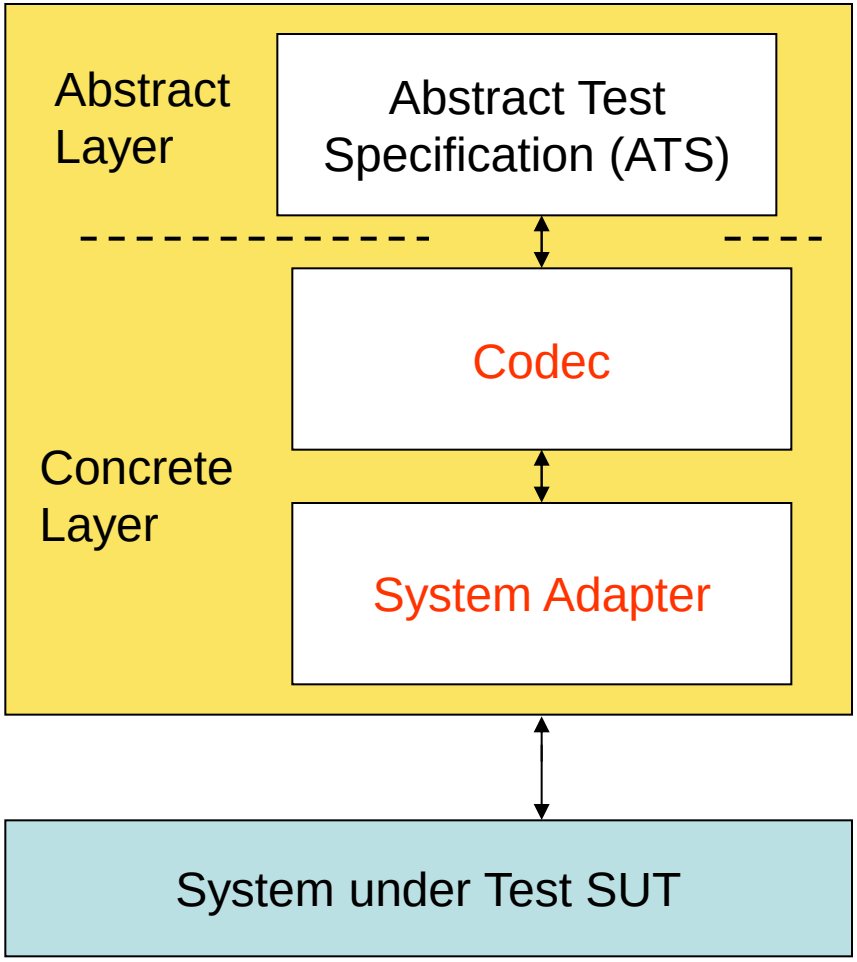
TTCN-3 Separation of Concerns : Abstract – Concrete Layer 1(2)



ATS a set of *modules* comprising

- test data: types of messages; type instances = *templates*
- test configurations: *ports* and *test components*
- test behavior: *functions, altsteps,* and *testcases*
- control: the global behavior of the test system

TTCN-3 Separation of Concerns : Abstract – Concrete Layer 2(2)



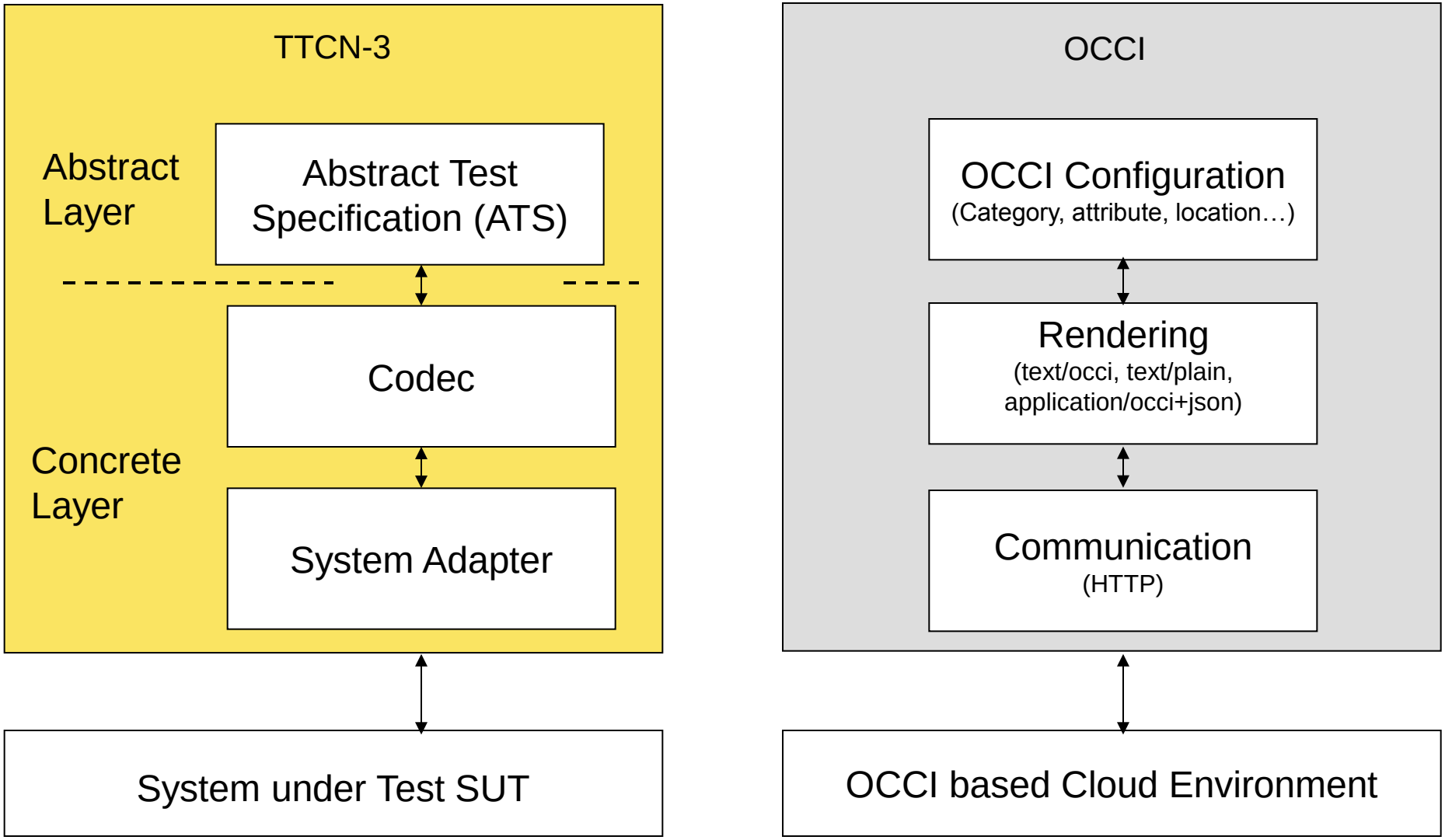
Codec

- "Rendering": MIME type data

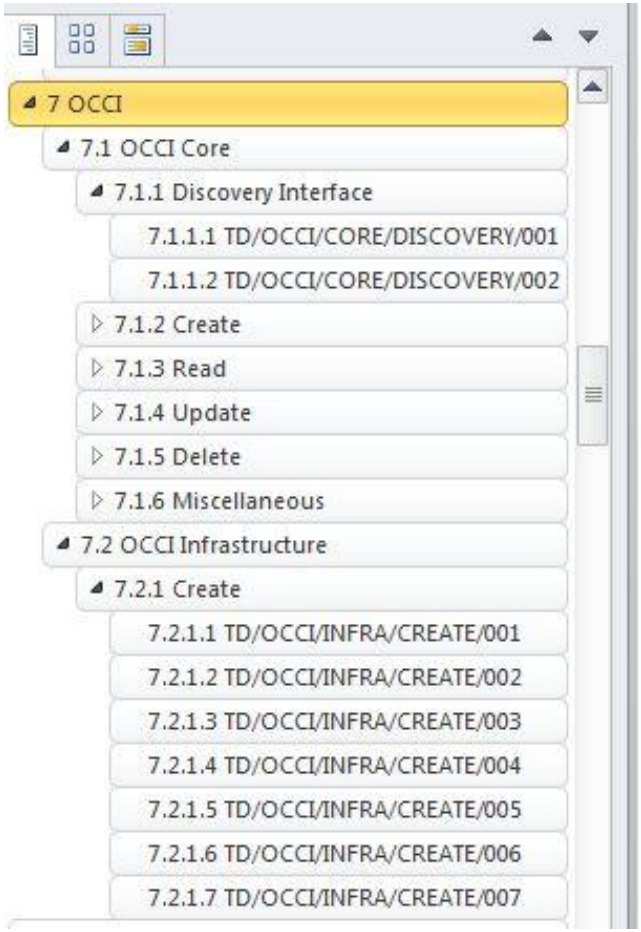
System Adapter

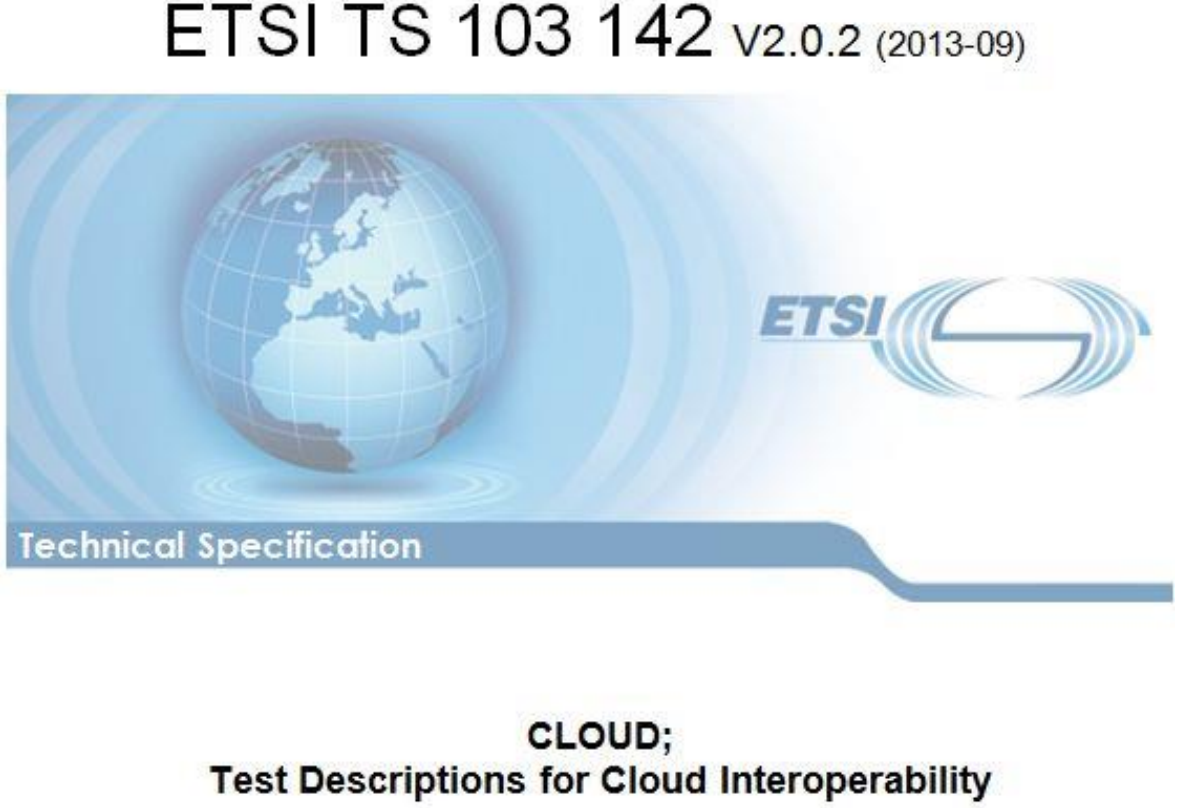
- Transport HTTP(S), AMQP, …

TTCN-3 - OCCI

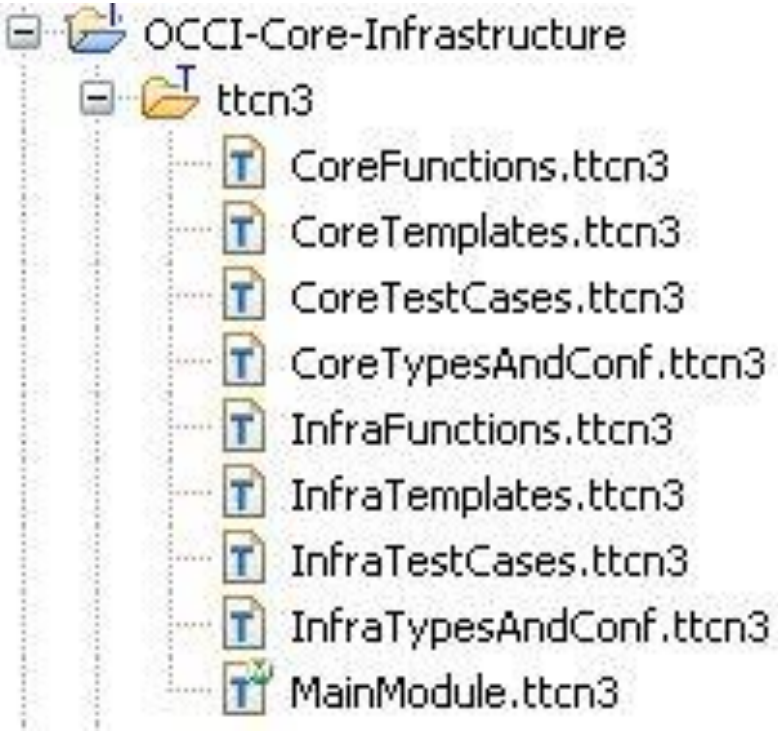

TTCN-3
Abstract Layer
Abstract Test Specification (ATS)
Codec
Concrete Layer
System Adapter
System under Test SUT
OCCI
OCCI Configuration
(Category, attribute, location…)
Rendering
(text/occi, text/plain, application/occi+json)
Communication
(HTTP)
OCCI based Cloud Environment

Test Descriptions for Cloud Interoperability – OCCI Part





Working towards TTCN-3 / OCCI 1(6)




```
modulepar charstring tenantname :="";
modulepar charstring username :="";
modulepar charstring password :="";
modulepar charstring kid :="";

/* select CRUD method */
modulepar boolean Create := false;
modulepar boolean Read := true;
modulepar boolean Update := false;
modulepar boolean Delete := false;

/* define HTTP headers*/
modulepar charstring X_Auth_Token:="";
modulepar charstring ContentType := "text/occi";
modulepar charstring AcceptValue := "text/plain";
```

Working towards TTCN-3 / OCCI 2(6)

```
if(Read){
    // Resource discovery
    execute (TD_OCCI_CORE_DISCOVERY_001());
    execute (TD_OCCI_CORE_DISCOVERY_002());

    // Resource Reading
    execute(TD_OCCI_CORE_READ_001());
    execute(TD_OCCI_CORE_READ_002());
    execute(TD_OCCI_CORE_READ_003());
    execute(TD_OCCI_CORE_READ_004());
    execute(TD_OCCI_CORE_READ_005());
    execute(TD_OCCI_CORE_READ_006());
}
if (Create) {
    // Resource Creation
    execute (TD_OCCI_CORE_CREATE_001());
    execute (TD_OCCI_CORE_CREATE_002());
    execute (TD_OCCI_CORE_CREATE_003());
    execute (TD_OCCI_CORE_CREATE_004());
    execute (TD_OCCI_CORE_CREATE_005());
    execute (TD_OCCI_CORE_CREATE_006());

    // Miscellaneous Functions
    execute (TD_OCCI_CORE_MISC_001());
    execute (TD_OCCI_CORE_MISC_002());
    execute (TD_OCCI_CORE_MISC_003());
}
```

```
if (Update){

    //Resource update
    execute(TD_OCCI_CORE_UPDATE_001());
    execute(TD_OCCI_CORE_UPDATE_002());
    execute(TD_OCCI_CORE_UPDATE_003());
}

if(Delete){
    //Resource deletion
    execute(TD_OCCI_CORE_DELETE_001());
    execute(TD_OCCI_CORE_DELETE_002());
    execute(TD_OCCI_CORE_DELETE_003());

    // Miscellaneous functions
    execute(TD_OCCI_CORE_MISC_004() );
}
```

Working towards TTCN-3 / OCCI 3(6)

```
module CoreTypesAndConf {

        type record URLReq {
             charstring protocol,
             charstring host,
             charstring port_number optional,
             charstring location,
             charstring resource_id optional,
             charstring actions optional
         }

        type charstring Attribute;

        type set of Attribute OCCIAttributes;

        type set OCCICategory {
             charstring term,
             charstring scheme,
             charstring class,
             charstring title optional,
             charstring relation optional,
             charstring location optional,
             OCCIAttributes attributes optional,
             charstring actions optional
        }

        type record OCCIRequest{
            URLReq url_req,
            OCCICategory occi_category optional
        }

        type set of OCCICategory OCCIServerResp;
```

Working towards TTCN-3 / OCCI 4(6)

- OCCI Mandatory Tests (TD/OCCI/CORE/DISCOVERY/001)
  - retrieving all OCCI Categories supported by the OCCI Server

Test Sequence:	Step	Type	Description
	1	**stimulus**	**OCCI Client requests all OCCI Categories supported by the OCCI Server**

	2	check	OCCI Client sends a HTTP GET request
			• Request-URI is /-/ or /.well-known/org/ogf/occi/-/
			• If HTTP Accept header is present it is containing at least one of the following MIME types:
			  • text/occi
			  • text/plain
			  • application/occi+json

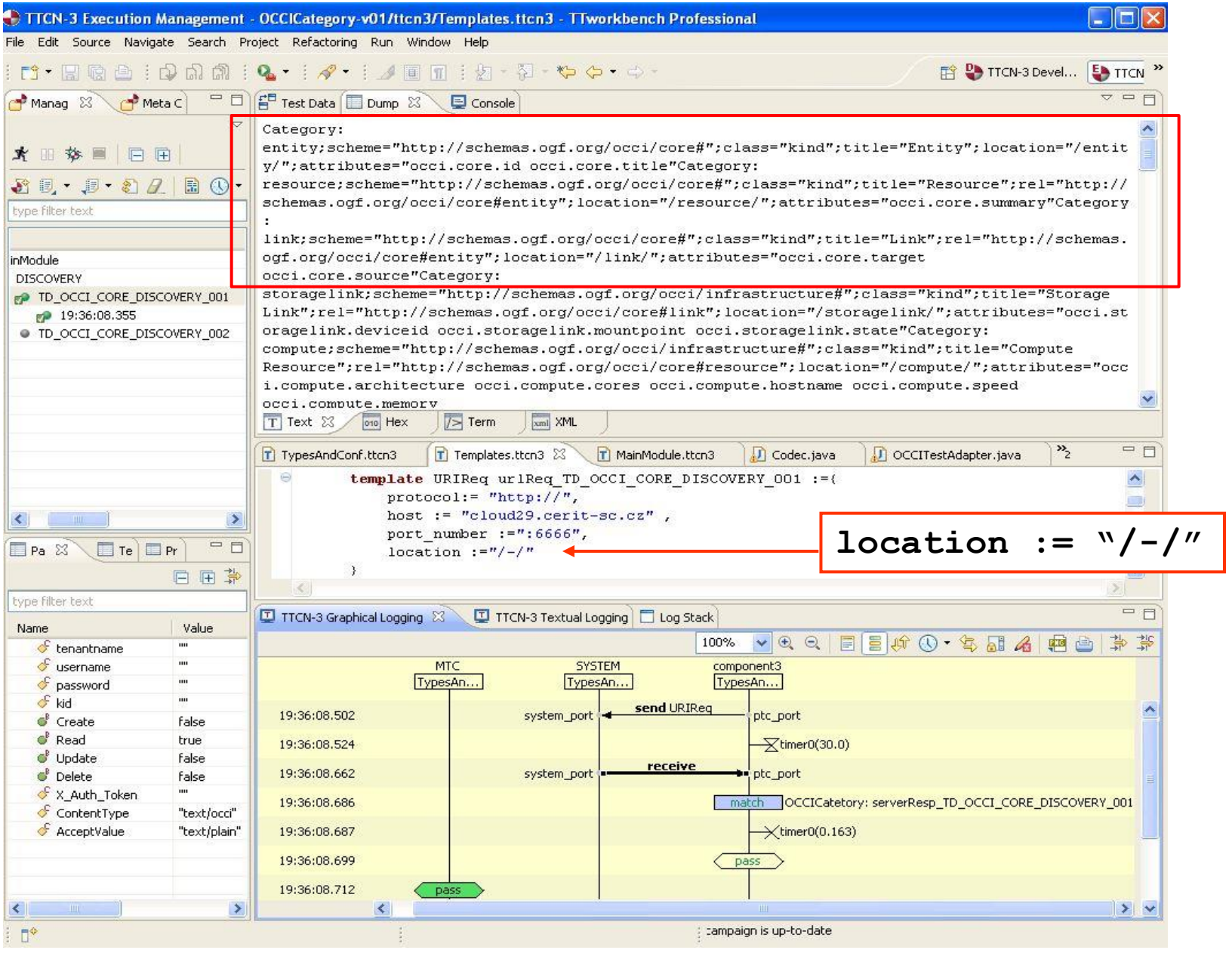
	3	check	OCCI Server sends a HTTP 200 (OK) response
			• HTTP Content-Type header corresponds to request's HTTP Accept header if present (see GDF.185 [3], clause 3.6.6)
			• HTTP Body contains all OCCI Categories supported by the OCCI Server and at least the following categories
			  • http://schemas.ogf.org/occi/core#entity
			  • http://schemas.ogf.org/occi/core#resource
			  • http://schemas.ogf.org/occi/core#link
			• The format of all OCCI Categories is compliant with the requested MIME type and the OCCI format restrictions

	4	**verify**	**OCCI Client displays the OCCI Categories received from the OCCI Server**

Working towards TTCN-3 / OCCI 5(6)

```
module CoreTestCases {

    import from CoreFunctions all;
    import from CoreTypesAndConf all;

        group DISCOVERY {

            testcase TD_OCCI_CORE_DISCOVERY_001() runs on MtcType system SystemType{

                        var PtcType ptc_discovery_001;

                        //create the PTC
                         ptc_discovery_001:= PtcType.create;

                        //map the PTC to the system port
                        map (ptc_discovery_001:ptc_port, system:system_port);

                        //start the PTC behaviour
                        ptc_discovery_001.start(f_TD_OCCI_CORE_DISCOVERY_001());

                        //wait for the PTC to terminate
                        ptc_discovery_001.done;
             }

           testcase TD_OCCI_CORE_DISCOVERY_002() runs on MtcType system SystemType{
```

Working towards TTCN-3 / OCCI 6(6)



Future Work

- Following the ETSI interoperability test descriptions to complete the test cases
- Structuring OCCI via TTCN-3 towards a maximum automaticity of the ETSI interoperability tests and
- Using other Cloud project results and the discussions in the OGF OCCI WG as guideline towards TTCN-3 - OCCI modeling of more complex cloud ecosystems

Acknowledgements

- Ina Schieferdecker; FU Berlin, Fraunhofer FOKUS
- TestingTechnologies, Berlin
- (partly) BonFIRE project, 7th Framework Program, grant agreement number 257386

Conclusion

- demonstrated initial work towards model driven testing of OCCI-based cloud systems using TTCN-3
- used the ETSI cloud interoperability document as initial guideline
- showed an OCCI/OpenNebula infrastructure as a candidate for the development of ‘ETSI-oriented’ TTCN-3/OCCI based interoperability tests
- proposed related further work

Questions?