IBM Engineering Lifecycle Management

Jazz Platform

Link Validity API

Link Discovery Services ldx

A Summary of capability - DRAFT

V0.3 19th Feb 2021

Gray Bachelor

Solution Architect

Development Ecosystem Team, IBM Engineering
IBM Cloud and Cognitive Software

Section 1: Table of Contents

Section 2: Introduction

This document provides a summary of two essential ELM link management services that apply when Configuration Management is enabled.

Link Validity API

Link Validity is primarily a manual marker to link status in particular contexts applies when Configuration Management is enabled, Link Validity also needs to be enabled for each ELM application server.

Link Discovery Services ldx

Link Discovery is an optional service for incoming link discovery to a given project area. Link discovery usage takes into account the ELM conventions for link storage across upstream and downstream apps, which is explained here.

This documents provides an overview and summary at V6.06.01 and V7.

Information provided as-is and is currently draft.

Usage outside of any ELM application is without warranty and may require additional licensing please consult IBM Offering Management. Usage by third parties is not yet officially support, again consult IBM.

Please read disclaimers in Appendix 4.

Section 3: Link Validity

IBM Engineering Lifecycle Management (ELM) suite applies Linked Data concepts using Open Services for Lifecycle Collaboration (OSLC), these concepts allow dependencies and associations to be made between certain types of artefacts in ELM, also with certain external non-ELM applications which use specific OSLC conventions. An application user may propose or define a relationship between artefacts using links but in general OSLC does not require or even elaborated use link properties, such as their status or applicability. In their most basic, and typical, form links are just information triples – link source URI, linktype URI and link target URI. Link Validity provides additional properties for visual and reportable indication of link status within ELM and is a common, aggregating utility provided by Jazz Foundation, which is a set of components and services found within the Jazz Team Server. In ELM users can manually set, unset and comment upon link validity states to indicate and track link validity status or approval. User link validity actions are the individual link or multi-select level.

Noting that OSLC does not define specific link validity mechanisms or properties, these emerged as customer requested that ELM provide similar support to DOORS classic link suspicion.

Noting further the role of OSLC Configuration Management which uses a containment paradigm rather than adding applicability properties to links. Links and Link validities are valid within a given configuration context, because they are bounded within a configuration container. There is no direct query on a link to find the configurations in which it is valid, a context is provided and a link status is returned, more of this later.

Implementation in ELM is based upon convention over apps “lifecycle position” – which assumes certain link ordering. There is a notion of links to upstream or downstream applications[1]:

Upstream apps “mature” or evolve earlier in the 1st run through of the lifecycle e.g. A Requirement

[1] Consult e.g. https://www.ibm.com/support/knowledgecenter/SSYMRC_7.0.3/com.ibm.jazz.vvc.doc/topics/c_linkval.html#c_linkval

Downstream apps “mature” or evolve later in the 1st run through of the lifecycle e.g. a test of some requirement

Changes typically propagate from upstream apps to downstream apps

Suppose that an artefact link exists, from a downstream app to an upstream app, such as a test validates some requirement. Suppose then that an upstream app user, e.g. in DNG, changes that requirement and then may mark the (incoming) link invalid. A downstream app user, e.g. in ETM, may then see that at least that test needs review or change to be able then to set the link validity back to valid. Which would be then reflected in the appearance and reporting on the link validity. Note that with Configuration Management enabled these changes and visibilities occur with specific Global or Local Configuration and Change Set contexts.

This convention applies to asymmetric link types, i.e. links resolved in downstream apps. A change to either end of a symmetric link with cause Link Validity to change.

A rolled-up Link validity summary is also provided according to directionality which is addressed below.

Link validity states

Link validity is analysed according to

Contents of artefact version X + link type + contents of artefact version Y => [has a] validity state

The link validity states supported are

Table 1 Link validity states

Link state	Meaning	Occurs	Can comments be added	App visibility
Unspecified / Unassigned	Unknown or not specified Assumed suspect	By default	No	ENI/RB
Valid	A human has marked as having asserted validity of the association between the artefacts, each in given state and a given configuration context	When a human as consciously marked a link as Valid		DNG/ETM/EWM ENI/R
Invalid	A human has marked as having asserted invalidity of the association between the artefacts, each in given state and a given configuration context	When a human as consciously marked a link as Invalid	Yes	DNG/ETM/EWM ENI/R
Suspect	ELM or a human has detected some change which may make an		Yes	DNG/ETM/EWM ENI/R[2]
Unknown	Expected Link Validity not available	A server may be down or an artefact deleted		ENI/R

[2] Suspect is not really a valid status – See ref

Link Validity Profile

Link Validity make use of an artefact Content Hash based upon a Validity Profile, the hash is created before or as the link status is set. Absence of the Content Hash for a given link can be taken as invalidity or suspicion, the latter is the default case in ELM apps.

The hash needs to be chosen (calculated) so as to have a low likelihood of being duplicated by another linking application, a strong cryptographic hash is used in the ELM apps The aim is to create the hash from a set of attributes which define the semantic content of the artefact itself, not artefact version marking, therefore such properties as version ID, version URL, creation time, modification time, contributor, etc. should normally be omitted from the hash calculation, as should the presence of links.

Applications should support the same content hash values from the same profile and from different profiles when the attribute values in question are the same; that reuse of the same content hash should be considered a valid case and not a 'collision'.

ELM currently supports only a default profile, and is not customisable. See Appendix 2

Link Validity information in the TRS Shape

There are essentially two records that appear in the JTS Link Validity TRS (2.0) around link validity, they must be combined for reporting:

Content Hash for Link Validity in TRS

The first of these records, the simpler, is a contentHash statement about a specific resource. As soon as the validity service is made aware of the validity of a particular resource version, this record is published in TRS. This is the configuration-independent yet artefact content specific mechanism that is used to truly determine if two versions of the same resource can be treated the same with respect to link validity.

An example of one of these Content Hash entries in RDF is as follows:

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:oslc="http://open-services.net/ns/core#"
   xmlns:validity="http://jazz.net/ns/validity#"
   xmlns:dcterm="http://purl.org/dc/terms/" >
   <rdf:Description rdf:about="https://rtpclmperf13-
rn.rtp.raleigh.ibm.com:9443/rm/resources/_53efdc5e9ea84b9e9b5b5eb2551e0748?revision=_DDYmDL-
2EeWGRdEY53LVLA">
      <validity:contentHash
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">90676a2d3f8c989f1cb3be1f00fc1053</validity:c
ontentHash>
   </rdf:Description>
</rdf:RDF>
```

Note that multiple versions of the same resource (although would have unique URIs) may align to the same content hash. This is expected and good.

Link Validity record in the TRS

The more complex item is for the link validity record itself. This has some basic metadata (who modified it, when, an optionally populated description).

The data needs to be unioned with the previously mentioned contentHash bits combined to provide the 'validity:status' for a given validity:validateTargetHash + validity:validateSourceHash + dcterm:type (representing link type) for the specific validity:profile.

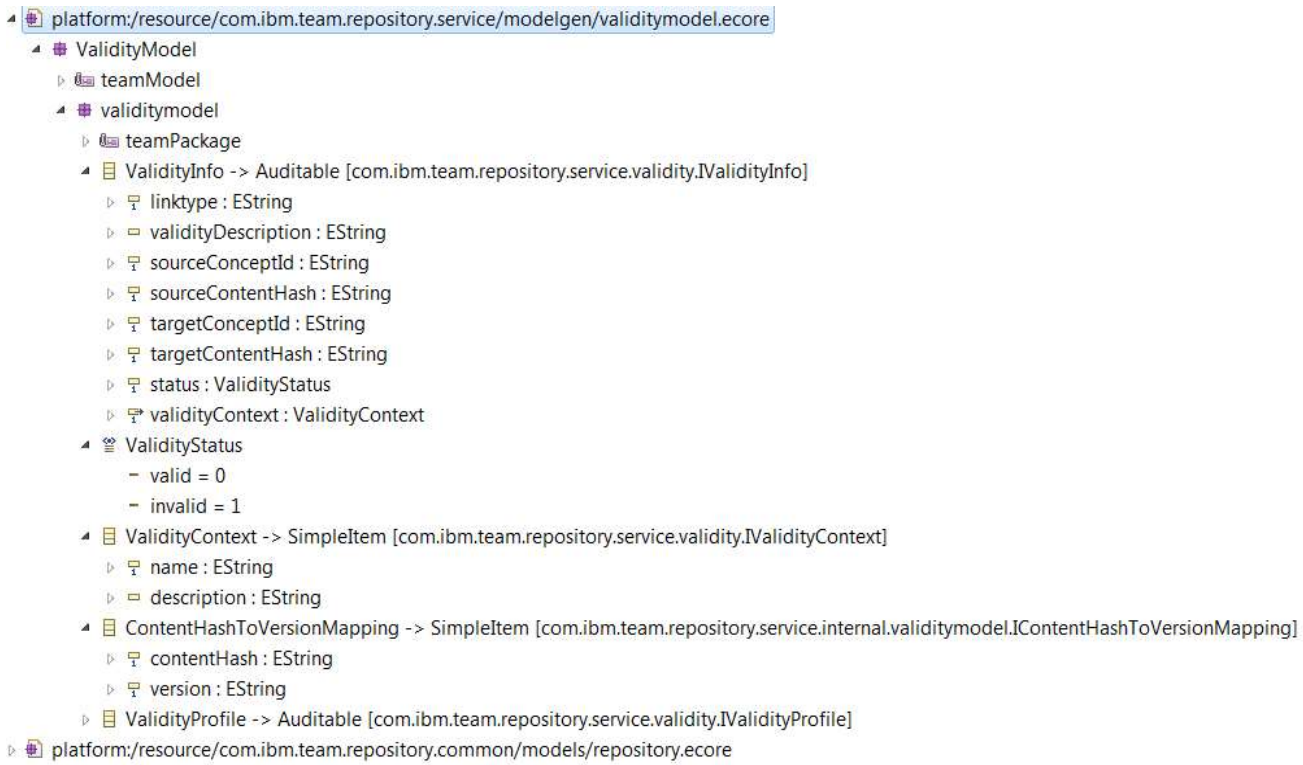
Currently in ELM a single profile is used.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:oslc="http://open-services.net/ns/core#"
  xmlns:validity="http://jazz.net/ns/validity#"
  xmlns:dcterm="http://purl.org/dc/terms/">
  <rdf:Description rdf:about="https://rtpclmperf13-rn.rtp.raleigh.ibm.com:9443/jts/validity/_QRksML-2EeWD7voD9vXaVQ">
    <validity:validateTargetHash rdf:datatype="http://www.w3.org/2001/XMLSchema#string">daa8e4118d774cb9129e2c0b42f3df46</validity:validateTargetHash>
    <dcterm:modified rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2016-01-20T20:42:00.186Z</dcterm:modified>
    <oslc:modifiedBy rdf:resource="https://rtpclmperf13-rn.rtp.raleigh.ibm.com:9443/jts/users/TestJazzAdmin1"/>
    <oslc:instanceShape rdf:resource="http://jazz.net/shapes/validity#ValidityResourceShape"/>
    <dcterm:description rdf:datatype="http://www.w3.org/2001/XMLSchema#string"></dcterm:description>
    <validity:validateSourceHash rdf:datatype="http://www.w3.org/2001/XMLSchema#string">90676a2d3f8c989f1cb3be1f00fc1053</validity:validateSourceHash>
    <validity:profile rdf:resource="https://rtpclmperf13-rn.rtp.raleigh.ibm.com:9443/jts/validity/_QBPxML-2EeWD7voD9vXaVQ"/>
    <rdf:type rdf:resource="http://jazz.net/ns/validity#LinkValidity"/>
    <dcterm:type rdf:resource="https://rtpclmperf13-rn.rtp.raleigh.ibm.com:9443/rm/types/_C0Qygb-2EeWGRdEY53LVLA"/>
    <validity:status rdf:resource="http://jazz.net/ns/validity#Valid"/>
  </rdf:Description>
</rdf:RDF>
```

A common report scenario would be something like 'show valid relationships in stream alpha'. The query output typical requirement or test case info (title, type, whatever else), but determine the result set based on the specified configuration(s), filtering through the contentHash data on based on the validity:status

Section 4: Link Validity API

Link Validity data model



Provided Service end points

On the ELM application side the REST APIs is available at

https://{appUrl}:{port}/{appContext}/validity

On the ELM JTS side the REST API is available at

https://{jtsUrl}:{port}/jtsContext}/validity

REST APIs

Provided services details are available at /validity?method=OPTIONS

The API only accepts or returns JSON objects

POST /validity/addValidity: add or update a list of validityInfo (valid or not). A validity link is updated if it as the same link type, source concept, source content hash, target concept, target content hash, and context name. Otherwise a new link is created

ValidityInfo: This is the object that is passed around to the validity service to add or update a link instance. The mandatory values: linktype, sourceConceptId, sourceContentHash, targetConceptId, targetContentHash, contextName.

A list of versionIds can also be passed to populate the contentHashToVersionmapping table at the same time during insertion of the validity link status.

See below for the properties names to use during JSON serialization

```
@BeanConstructor(properties={"linkType","validityDescription","sourceConceptId","sourceContentHash",
        "targetConceptId","targetContentHash","status","contextName", "contextDescription",
        "sourceVersionIds", "targetVersionIds"})
public ValidityInfo(String linkType, String validityDescription,
                    String sourceConceptId, String sourceContentHash, String targetConceptId,
                    String targetContentHash, int status, String contextName,
                    String contextDescription, String[] sourceVersionIds, String[] targetVersionIds)
```

POST /validity/addVersionsMapping: For a given content hash, add a list of versions to the ContentHashToVersionMapping table. These new versions will then be associated to the links which reference this content hash.

VersionMapping : This is the object that is passed around to the validity service map a list of versions to a contentHash. See below for the properties names to use during JSON serialization

```
@BeanConstructor(properties={"contentHash","versionIds"})
public VersionsMapping(String contentHash, String[] versionIds)
```

POST /validity/isValid: Verify if a link is valid given a link type, a source concept, a source content hash, a target concept, a target content hash and a context name. If not found, 404 is returned.

IsValidQuery : This is the object that is passed around to the validity service to verify the validity of a link. See below for the properties names to use during JSON serialization

```
@BeanConstructor(properties={"linkType","sourceConceptId" , "sourceContentHash", "targetConceptId",
        "targetContentHash", "contextName"})
public IsValidQuery(String linkType, String sourceConceptId, String sourceContentHash,
        String targetConceptId, String targetContentHash, String contextName) {
```

POST /validity/validTargetVersions: Return the list of valid target versions given a link type, the source concept, source content hash, the target concept and context name. 404 is returned if no links found

QueryValidTargetVersions : This is the object that is passed around to the validity service. See below for the properties names to use during JSON serialization

```
@BeanConstructor(properties={"linkType","sourceConceptId", "sourceContentHash", "targetConceptId",
        "contextName"})
public QueryValidTargetVersions(String linkType, String sourceConceptId, String sourceContentHash,
        String targetConceptId, String contextName) {
```

POST /validity/validSourceVersion: Return the list of valid target versions given a link type, the source concept, the source content hash, the target concept and the context name. 404 is returned if no links found

QueryValidSourceVersions : This is the object that is passed around to the validity service. See below for the properties names to use during JSON serialization

```
@BeanConstructor(properties={"linkType","sourceConceptId","targetConceptId","targetContentHash",
        "contextName"})
public QueryValidSourceVersions(String linkType, String sourceConceptId,
        String targetConceptId, String targetContentHash,
        String contextName) {
```

Link Validity API examples

Accessing the APIs

Rich client can authenticate and access the REST APIs using IteamRawRestServiceClient

```
RemoteTeamServer server = (RemoteTeamServer) TeamServerFactory.INSTANCE.newTeamServerFromURL(getJazzServerUrl(),
        LenientCertificateValidator.INSTANCE);
server.setCredentials("ADMIN", "ADMIN");
final ITeamRawRestServiceClient rawRestServiceClient = new TeamRawRestServiceClient(server);
```

Or

```
if(! TeamPlatform.isStarted())
    TeamPlatform.startup();
ITeamRepository repo = TeamPlatform.getTeamRepositoryService().getTeamRepository("https://localhost:9443/jazz");
if(!repo.loggedIn()) {
    repo.registerLoginHandler(new ITeamRepository.ILoginHandler() {
         public ILoginInfo challenge(ITeamRepository repository) {
             return new ILoginInfo() {
                 public String getUserId() {
                     return "ADMIN";
                 }

                 public String getPassword() {
                     return "ADMIN";
                 }
             };
         }
     });
     try {
        repo.login(null);
    } catch (TeamRepositoryException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
final ITeamRawRestServiceClient rawRestServiceClient = repo.getRawRestServiceClient();
```

Create/add a link validity record using addValidity

```
con = rawRestServiceClient.getConnection(new URI(getJazzServerUrl() + VALIDITY_URL + "/addValidity").normalize());
con.addRequestHeader("Accept", "text/json");


String content = "[" +
        "{" +
        "linkType:t1," +
        "validityDescription:t1," +
        "sourceConceptId:t1," +
        "sourceContentHash:t1," +
        "targetConceptId:t1," +
        "targetContentHash:t1," +
        "status:0,"+
        "contextName:t1," +
        "sourceVersionIds:[\"sv1\",\"sv2\"]," +
        "targetVersionIds:[\"tv1\",\"tv2\"]," +
        "}" +
        "]";

Response resp = con.doPost(new ByteArrayInputStream(content.getBytes("UTF-8")), content.length(), "text/json");
assertTrue(resp.getStatusCode() == HttpStatus.SC_OK);
```

Add a contentHash for a specific artefact version using addVersionsMapping

```
con = rawRestServiceClient.getConnection(new URI(getJazzServerUrl() + VALIDITY_URL + "/addVersionsMapping").normalize());
con.addRequestHeader("Accept", "text/json");
String mapping =  "{" +
        "contentHash:t1," +
        "versionIds:[\"t1\"]}";
Response resp = con.doPost(new ByteArrayInputStream(mapping.getBytes("UTF-8")), mapping.length(), "text/json");
assertTrue(resp.getStatusCode() == HttpStatus.SC_OK);
```

Sets a link as valid using isValid

```
con = rawRestServiceClient.getConnection(new URI(getJazzServerUrl() + VALIDITY_URL + "/isValid").normalize());
con.addRequestHeader("Accept", "text/json");

String content = "{" +
                        "linkType:t1," +
                        "sourceConceptId:t1," +
                        "sourceContentHash:t1," +
                        "targetConceptId:t1," +
                        "targetContentHash:t1," +
                        "contextName:t1" +
                        "}";

Response resp = con.doPost(new ByteArrayInputStream(content.getBytes("UTF-8")), content.length(), "text/json");
assertTrue(resp.getStatusCode() == HttpStatus.SC_OK);
Boolean r = JsonParser.DEFAULT.parse(new InputStreamReader(resp.getResponseStream(), Charset.forName("UTF-8")), -1, Boolean.class);
assertTrue(r);
```

Checking for validSourceVersions

```
con = rawRestServiceClient.getConnection(new URI(getJazzServerUrl() + VALIDITY_URL + "/validSourceVersions").normalize());
con.addRequestHeader("Accept", "text/json");

String content = "{" +
                        "linkType:t1," +
                        "sourceConceptId:t1," +
                        "targetConceptId:t1," +
                        "targetContentHash:t1," +
                        "contextName:t1" +
                        "}";

Response resp = con.doPost(new ByteArrayInputStream(content.getBytes("UTF-8")), content.length(), "text/json");
assertTrue(resp.getStatusCode() == HttpStatus.SC_OK);
JSONArray jObj = JSONArray.parse(new InputStreamReader(resp.getResponseStream(), HttpConstants.DEFAULT_ENCODING));
assertTrue(jObj.contains("t1"));
```

Setting contentHash for an artefact version and Checking for validTargetversions

```
con = rawRestServiceClient.getConnection(new URI(getJazzServerUrl() + VALIDITY_URL + "/addVersionsMapping").normalize());
con.addRequestHeader("Accept", "text/json");
String mapping =  "{" +
        "contentHash:t1," +
        "versionIds:[\"t1\"]}";
Response resp = con.doPost(new ByteArrayInputStream(mapping.getBytes("UTF-8")), mapping.length(), "text/json");
assertTrue(resp.getStatusCode() == HttpStatus.SC_OK);

con = rawRestServiceClient.getConnection(new URI(getJazzServerUrl() + VALIDITY_URL + "/validTargetVersions").normalize());
con.addRequestHeader("Accept", "text/json");

String content = "{" +
                        "linkType:t1," +
                        "sourceConceptId:t1," +
                        "sourceContentHash:t1," +
                        "contextName:t1," +
                        "targetConceptId:t1" +
                        "}";

resp = con.doPost(new ByteArrayInputStream(content.getBytes("UTF-8")), content.length(), "text/json");
assertTrue(resp.getStatusCode() == HttpStatus.SC_OK);
JSONArray jObj = JSONArray.parse(new InputStreamReader(resp.getResponseStream(), HttpConstants.DEFAULT_ENCODING));
assertTrue(jObj.contains("t1"));
```

Client API

IValidityClientService

This is the service which applications should use

IValidityClientService can be access as 0.6(?) services or through REST API

IValidityClientService can be accessed on either the application or the JTS side

By default the application validity uses the connected JTS as a validity provider

The applications just forwards the calls to the JTS IValidityClientService

The used validity provider can be changed in the server “Advanced server properties”

Section 5: Link Discovery Service

The ELM Link Discovery Service[3] ldx is a specialised version of the ELM Lifecycle Query Engine (LQE) and so uses the same technology to aggregate link information across the ELM applications and other registered data sources. It uses the same OSLC Tracked Resource Set feeds as LQE. Its purpose is to provide a common service for incoming link discovery when Configuration Management is enabled. By default ldx updates every 60 secs, noting that the default for GC cache update is 5 secs[i]. Within ELM on screen its named as Link Index Provider

Because it's a specialised version of LQE, ldx can be moved to its own server[4] and is a feed to ENI and Reporting.

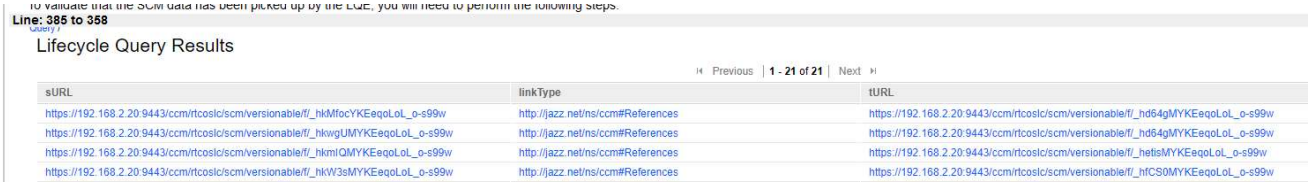
Link storage conventions across ELM are summarised as

Application	Stores links to	Queries for incoming links from
RM (DOORS Next)	RM	AM, CCM, QM
AM (RMM)	AM, RM	CCM, QM
QM (ETM)	QM, AM, RM	CCM
CCM (EWM)	CCM, QM, AM, RM	

ELM applications may use ldx or OSLC Query for incoming links. LDX contains information on all links and all configurations, so can do a configuration scoped search for links to a given artefact in a given configuration (global or local).

Sample Query result

(Placeholder)



[3] Some jaz.net articles refer to Link Index Provider

[4] https://jazz.net/wiki/bin/view/Deployment/DistributedLDX superceded by https://www.ibm.com/support/pages/node/6348196

Section 6: Change Logs

Major changes

Date / Version	Change	Who	Notes
V0.1	Format and create	Gray Bachelor	Link Validity pre-final
V0.2	Formatting and updates	Gray Bachelor	Out for review
V0.3	Add ELM arch overview	Gray Bachelor	Share draft with partner

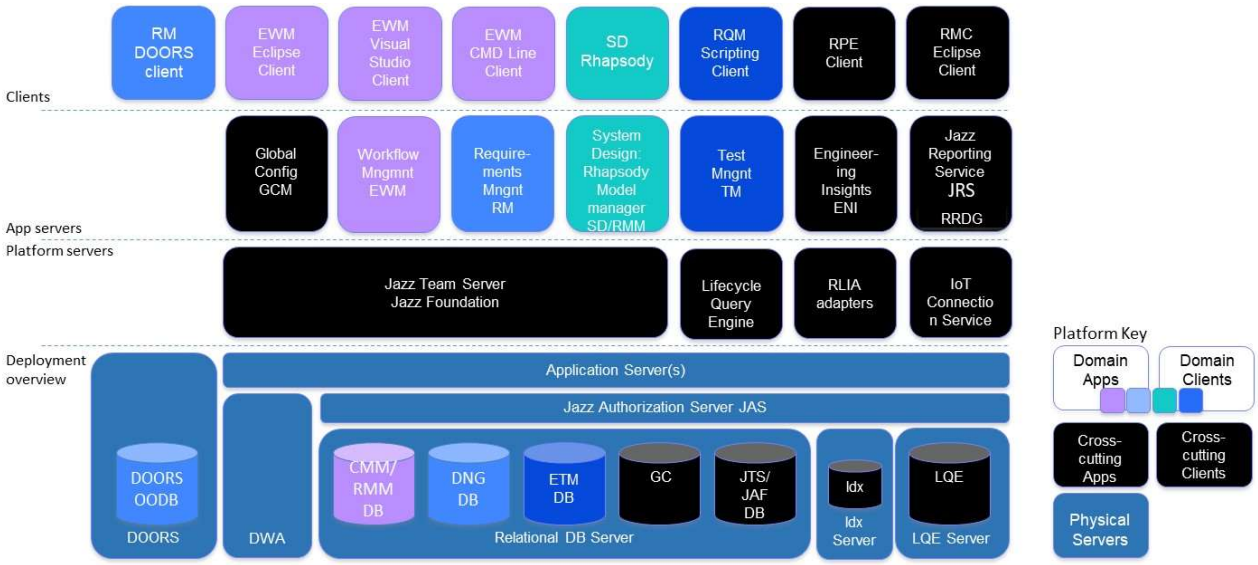
Major issues

Date / Version	Issue	Action due	Notes
V0.3	Lack ldx detail		

Docs used

Date / Version	Document	Notes

Appendix 1: Architectural Context



Link Validity is provided by the Jazz Team Server / Jazz Foundation server

Link Discovery / Index Provider is a separate server

Appendix 2: Current ELM Validity profile

Since ELM 6.0.1 profile behaviour is fixed and unmodifiable

Behaviour is outlined below as per which artefact attributes and link directionality will be acknowledged in this standard behaviour prior to the introduction of fully user configurable profiles.

DNG Artefacts attributes considered for validity:

DNG will include all user modifiable fields since ELM 6.0.1

Primary Text, Title, Description and Custom-defined attributes

QM Test Cases attributes considered for validity:

<Missing>

RMM SCM and Architecture artefact properties considered for validity:

<Missing>

DM Artefacts attributes considered for validity:
DM uses the same artefact attributes used to compute their eTags.

(Nearly identical behaviour to DNG)

Link types (directionality) for validity:

This directionality will only effect 'validity summary' (rollup)

Source** (affects validity)	A to B	Target (does not affect validity)
Source (does not affect validity)	B to A	Target** (affects validity)

Category	Source	LinkType	Target	Tracked since 6..0.1	Comments
OSLC Links					
OSLC (RM)	Requirement**	Elaborated By	Use case	No	
OSLC (RM)	Use case	Elaborates	Requirement**	No	
OSLC (RM)		Specified By	Requirement**	No	
OSLC (RM)	Requirement**	Specifies		No	
OSLC (RM)	WorkItem**	Affects	Requirement	No	
OSLC (RM)	Requirement	Affected By	WorkItem**	No	
OSLC (QM)	Test Case	Validates	Requirement**	**Yes**	
OSLC (QM)	Requirement**	Validated By	Test Case	**Yes**	

Category	Source	LinkType	Target	Tracked since 6..0.1	Comments
OSLC (CCM)	Change Request	Tracks	Requirement**	No	
OSLC (CCM)	Requirement**	Tracked By	Change Request	No	
OSLC (CCM)	Change Request	Implements	Requirement**	No	
OSLC (CCM)	Requirement**	Implemented By	Change Request	No	
OSLC (RM)	Requirement**	References	Requirement**	**Yes**	
OSLC (RM)	Requirement**	Referenced By	Requirement**	**Yes**	
OSLC (DM)	Requirement**	Derives	Architecture Element	**Yes**	
OSLC (DM)	Architecture Element	Derives From	Requirement**	**Yes**	
OSLC (DM)	Architecture Element	Satisfies	Requirement**	**Yes**	
OSLC (DM)	Requirement**	Satisfied By	Architecture Element	**Yes**	
OSLC (DM)	Architecture Element**	Trace	Requirement**	**Yes**	
OSLC (DM)	Requirement**	Traced by	Architecture Element**	**Yes**	
OSLC (DM)	Architecture Element	Refines	Requirement**	**Yes**	
OSLC (DM)	Requirement**	Refined By	Architecture Element	**Yes**	
RM Internal Links					
RM	Requirement**	Link To	Requirement**	**Yes**	
RM	Requirement**	Link From	Requirement**	**Yes**	
RM	Requirement**	Parent Of	Requirement**	**Yes**	
RM	Requirement**	Child Of	Requirement**	**Yes**	
RM	Requirement**	Synonym	Requirement**	**Yes**	
RM	Requirement**	Embeds	Requirement**	**Yes**	

Category	Source	LinkType	Target	Tracked since 6..0.1	Comments
RM	Requirement**	Embedded in	Requirement**	**Yes**	
RM	Requirement**	Extracts	Requirement**	**Yes**	
RM	Requirement**	Extracted from	Requirement**	**Yes**	
RM	Requirement**	<Custom link type>	Requirement**	**Yes**	
DM Internal Links					
DM	Architecture Element**	Links to	Architecture Element**	**???**	
DM	Architecture Element**	<Custom link type>	Architecture Element**		

Appendix 3: Links to additional info

Introduction to Link Validity

https://www.ibm.com/support/knowledgecenter/SSYMRC_7.0.3/com.ibm.jazz.vvc.doc/topics/c_linkval.html#c_linkval

Administer Link Validity across ELM servers

https://www.ibm.com/support/knowledgecenter/SSYMRC_7.0.3/com.ibm.jazz.vvc.doc/topics/t_linkval_enbl.html

Reporting on Link validity

https://www.ibm.com/support/knowledgecenter/SSYMRC_7.0.3/com.ibm.jazz.vvc.doc/topics/t_linkval_report.html#t_linkval_report

Additional useful info not linked

Andy Lapping's Link Existence and validity enablement presentation for V7

Appendix 4: Disclaimers and notices

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.
Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

[i] https://www.ibm.com/support/knowledgecenter/SSYMRC_6.0.6.1/com.ibm.jazz.vvc.doc/topics/c_cm_cfg-settings.html#c_cm_cfg-settings