#TXS2020

SDL

Tridion Docs Architectural Runway

Dave De Meyer

10 December 2020

USER Day 2012

SDL LiveContent 2012 Bootcamp on Security & What is New in Architect 2012 (10.0)

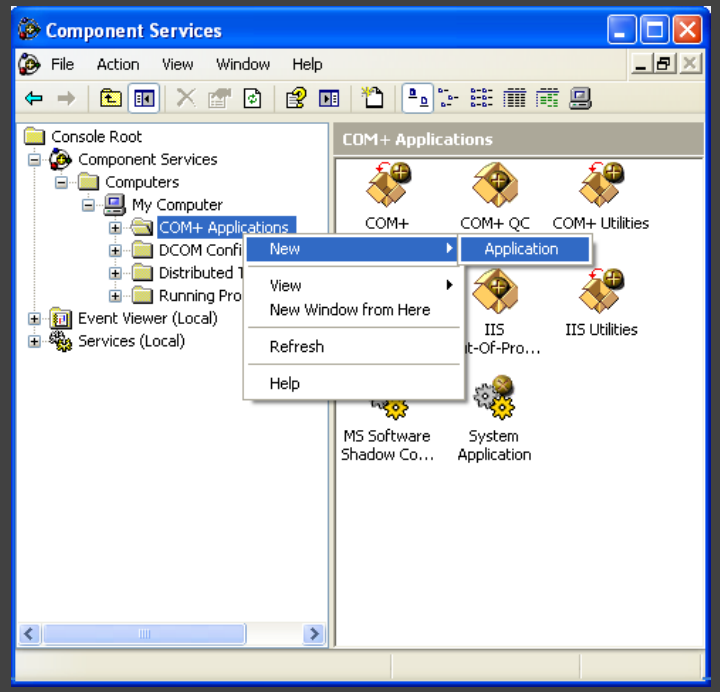
Dave De Meyer, Development Manager

So much to tell…

- I will stick to the Tridion Docs Content Manager server-side
- Touching integrations, Content Delivery, and Client Tools (like Publication Manager)

- I will share three trains-of-thought; for each covering
  - History
  - Now
  - Next
  - Enablers

- Disclaimer: Subject to change, we acquire new insights as we go
  - The current road book we are executing on



.NET Core

Microsoft® .NET

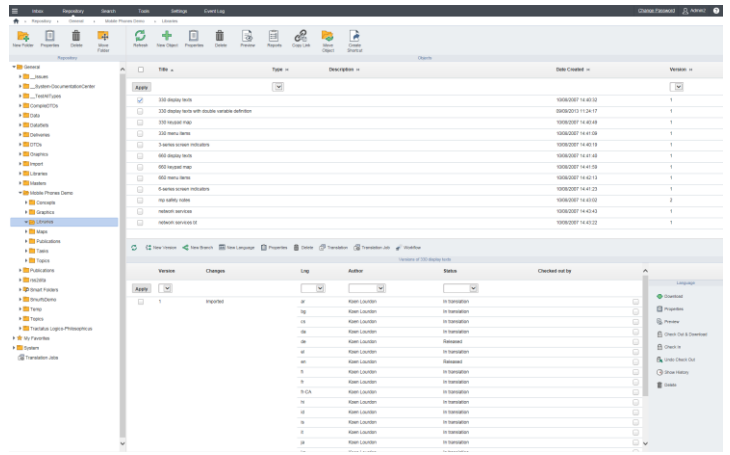
.NET

SDL*

History – From COM+ tech into .NET Framework

- Examples on vertical, so functional, migration across releases
  - Faster API25 functions allowing **.NET based IWrite*plugins** instead of COM+based IOnDocStore plugins
  - Migrated from Microsoft Message Queue (MSMQ) into **BackgroundTask** which can handle 1s to 24h+ operations
  - **Translation Management** revisited, enabling Translation Organizer tight integrations to WorldServer, TMS,…
  - **Publish rewrite**; faster, more extension points and better recovery
  - Etc
- Sharing the mantra of ‘same or better performance’

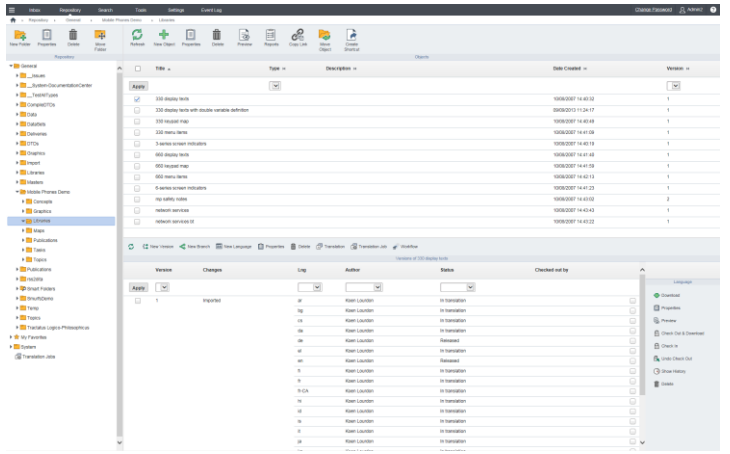
Now – From COM+ tech into .NET Framework

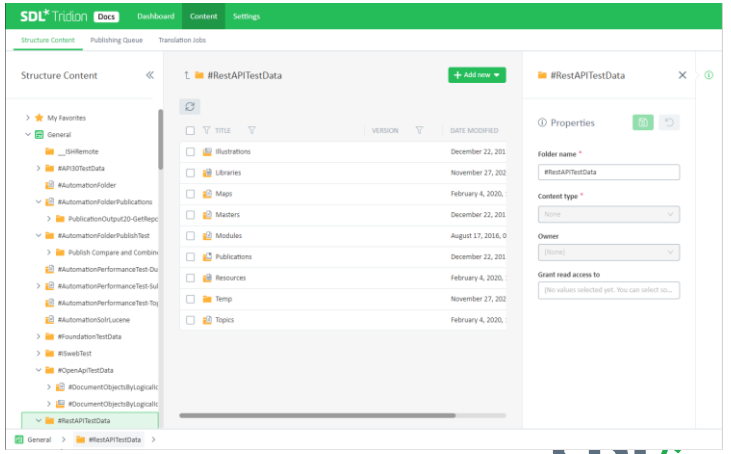


- ‘You are here’
  - Two data layers, one behind the Web Client (ISHCM) and one behind all the rest. Needs maintenance across Oracle RDBMs and Microsoft SQLServer versions
  - Dropped the managed C++ runtime requirement
  - Recent release the COM+ data layer is reduced to read-operations only. Reducing our Microsoft Distributed Transaction Coordinator (MSDTC) to install time only.
  - COM+ - so DLL Hell – we retreated from the Global Assembly Cache (GAC) on the server except for one COM+ interop assembly
- Web Client is built around the ClassicASP / COM+ combination. Evolving is too difficult; functionally but also regarding security

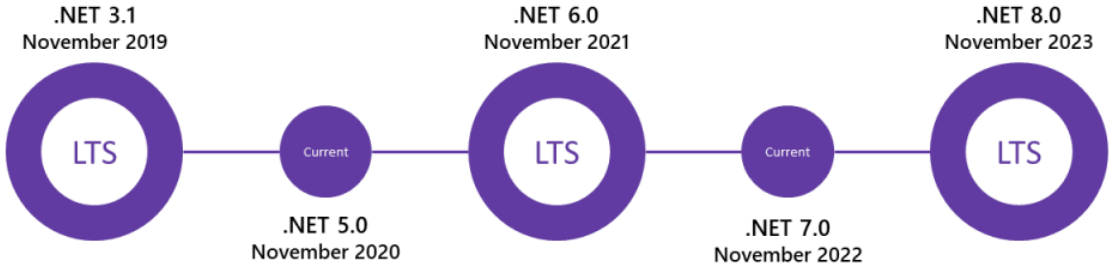
Next – From COM+ tech into .NET Framework

- Web Client is built around the ClassicASP / COM+ combination and ASP.NET
- **Evolution happened** resulting in
  - Properties, Settings, EventLog, TranslationJob, etc
- **Evolving is too difficult**; one-UX, functionally but also regarding security
  - Repository, Inbox, Reporting, Search, Condition Management, etc will be revisited
  - Too big for one-stop shopping; focus on personas like Administrators and Translation Coordinators
  - Enabling you through minimally one of our user interfaces





Next – From .NET Framework into .NET (Core)



- Microsoft announced that .NET Framework 4.8 is the last version of the full framework
  - Still patches and aligned with Operating System Life Cycle
  - Windows only, similar to PowerShell 5.1
- **Microsoft .NET** (Core)
  - Cross platform, with Windows compatibility packages (e.g. Registry Keys)
  - Supports PowerShell 7 (i.e. no WS-Trust support in .NET Core means no ISHRemote on PowerShell 7 on Linux)
- Microsoft announced revisited **support policy** of NET (Core)
  - .NET Core 3.1 is a Long-Term-Service (LTS) release
  - .NET Core 4.x is skipped to avoid confusion with widely distributed .NET Framework 4.x
  - .NET 5 (dropping Core) is a Short-Term-Service release
  - .NET 6 is an LTS, natural successor of .NET Core 3.1
- Remember the people angle as preferred **tooling support** like Visual Studio Code affects productivity

Enablers – From COM+ tech into .NET (Core)

- Our and custom Plugins/Handlers are advised to be written in **.NET Standard 2.0**
  - This way they work on .NET Framework 4.7.2/4.8.0 but also on .NET (Core) 3.1
- Save guards the product as **new developers** don't graduate on .NET Framework or ClassicASP
  - People working on the product, with the product, for partners, for you
- **Microsoft Investment** on next generation .NET (Core)
  - Lessons learned, patterns, security, performance happens here
- One data layer allows **database model refactoring**, again higher throughput

Web Services – ASMX and SVC

- ASMX based web services like http://.../InfoShareWS/Application.ASMX
  - Since 2003
  - First parameter in every function is always ‘AuthenticationContext’, so the Trisoft way of authentication
- Introducing Windows Communication Foundation (WCF) services like http://.../InfoShareWS/WCF/API25/Application.SVC
  - Support for claims-based authentication
  - Replaces ASMX Web Services, so marking them as deprecated
    - Deprecated here means supported as long as the cost of maintenance is reasonable
    - Goal is to step away from Trisoft Authentication (Trisoft Username/Password combinations)

44

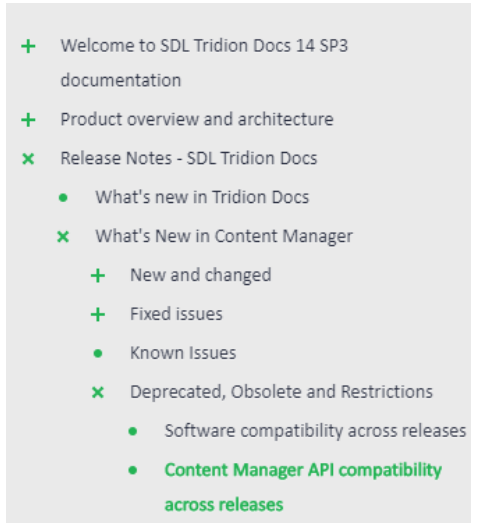
History – From API 2.0 to API 2.5

- Actually only difference between **API 1.0** and **2.0** is that the version parameter went from a Long to a String.
  - Allowing branched versions (e.g. “3.1.5”) and filter keywords (e.g. “New”, “Latest 3.1”)
- The API versioning of **2.0** to **2.5** was deliberate, rewriting from COM+ to .NET Framework with functional backward compatibility in mind
  - Technically you still need to rewire code from e.g. DocumentObj**20** to DocumentObj25
  - We just kept the function definitions very close, and the WS-Trust security needed no changes
- Remember that Client Tools, Translation Organizer, ISHRemote,… are all **build on that same public API**
  - Available to you!
  - We share your concern!
- Sharing the mantra of ‘make it better not different’

Now – From API 2.0 to API 2.5

API Versioning across product releases

- Holding on to 'D'eprecated API calls as long as cost allows
  - Remember COM+ to .NET, only read-operations (MSDTC), database model changes, etc
- Providing alternatives, based on our experience
- Still some 'S'upported API 2.0 calls without 2.5 alternative

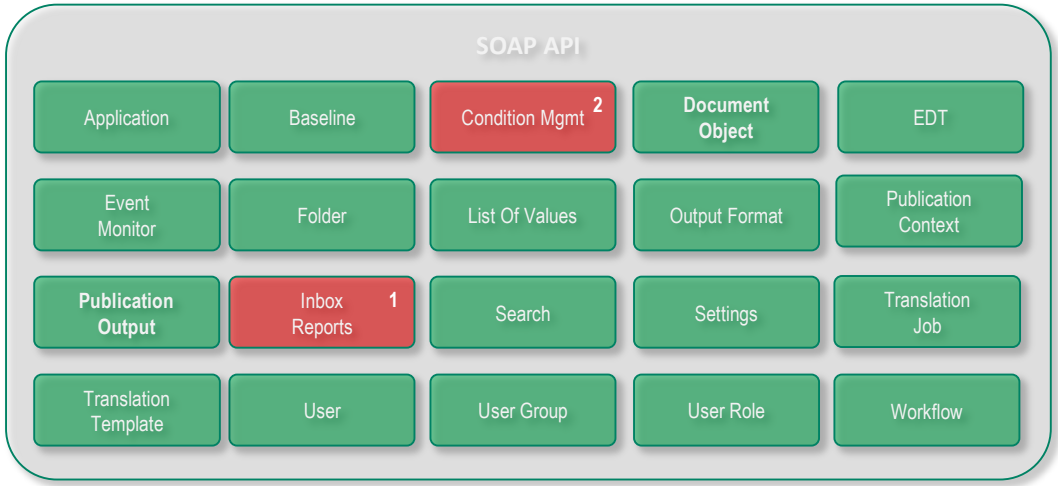


Method	10.0.X / 2013	11.0.X / 2014	12.0.X / 2016	13.0.X / 13	14.0.X / 14	Replaced By...
API25.Annotation.Create	-	-	-	-	S	
API25.Annotation.CreateReply	-	-	-	-	S	
API20.Reports.GetReferencedByDocObj	D	D	D	-	-	API25.DocumentObj.ContainingLogicalId
API20.Reports.GetReferencedDocObj	S	S	S	D	D	
API20.Reports.GetReferencesByIshLngRef	S	S	S	D	D	API25.DocumentObj.GetChildren (using multiple calls per LinkType)
API25.Search.PerformSearch	S	S	S	S	S	
API25.Search.PerformSearchInPublication	-	S	S	S	S	
API20.Settings.GetSystemLanguage	D	D	D	-	-	API25.Settings.GetMetadata using the field FMASTERLNG
API20.Settings.GetSystemResolution	D	D	D	-	-	API25.Settings.GetMetadata using the field FISHSYSTEMRESOLUTION
API25.UserRole.Update	S	S	S	S	S	
API20.WorkFlow.GetInboxContent	S	S	S	S	S	
API20.WorkFlow.GetInboxes	S	S	S	S	S	
API20.WorkFlow.PerformAction	S	D	D	D	-	

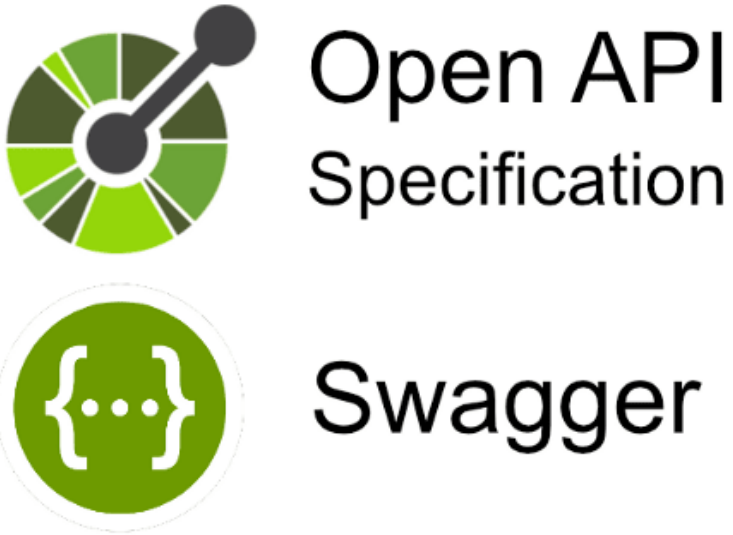
Source: https://docs.sdl.com/806356/791434/sdl-tridion-docs-14-sp3/content-manager-api-compatibility-across-releases

Now – From API 2.0 to API 2.5

- **Full stack .NET Framework**; from client over business over plugins into the database
- Remember Web Client goes over COM+, actually Web Services still has two 2.0 calls
  - Inbox20
  - ConditionMgmt20, especially Synchronize
- Both WCF-SOAP and ASMX-SOAP are communication-protocol and security-protocol antennas on the API25 .NET assembly layer
  - API25 assembly for controlled In-Process usage in IWrite-plugins and more



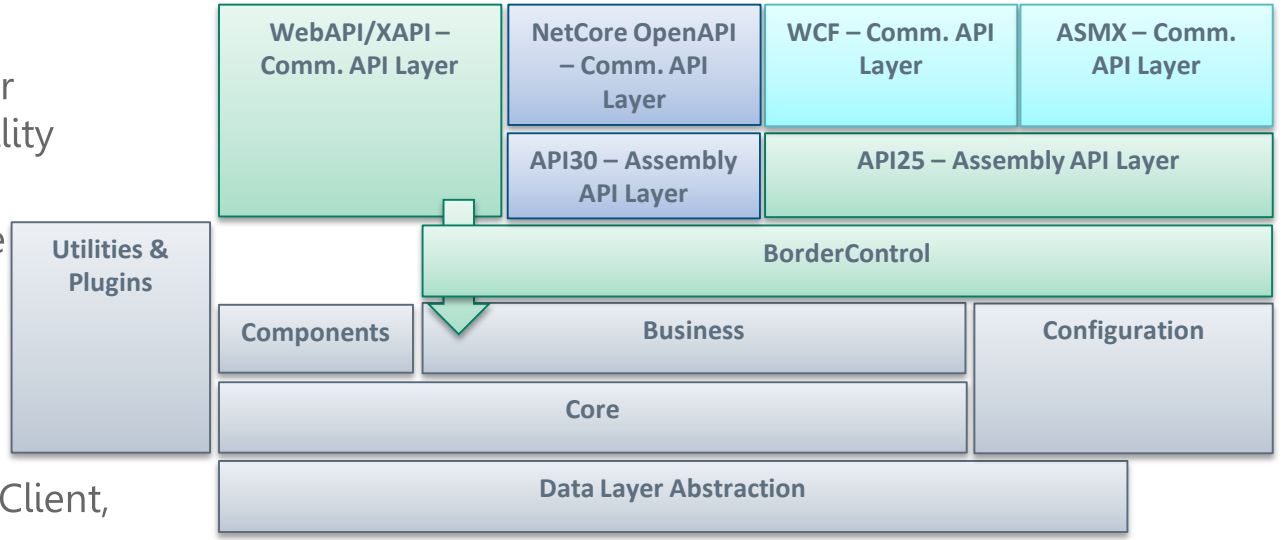
Next – From API 2.5 to API 3.0



- **Communication-protocol changes to OpenAPI (aka Swagger)**
  - **RESTful APIs** over HTTP protocol
  - Data representation using **JSON** instead of XML
  - Great **interoperability**, also for non-NET languages through available client generators
  - Using HTTP Verbs, resource-oriented, **HATEOS** (think permissions) and more
  - OpenAPI (Swagger UI) developer **documentation**
  - **Partial read/write** by continuing support for 'RequestedMetadata' and more to avoid fully loaded objects

Next – From API 2.5 to API 3.0

- Layering, new **API30** assembly with explicit **models** to support WebAPI
- **Internal/Private API surface: v0**
  - Obsoletes private WebAPI/XAPI layer
  - Same quality, supporting Organize Space successor
  - No guarantees on cross product version compatibility
- **Public OpenAPI surface: v3, so API 3.0**
  - Obsoletes ASMX-SOAP and WCF-SOAP API surface
  - Business compatibility is there
  - Rewiring required. We share your concern!
- Ball park sizing
  - 170+ WebApi/XAPI calls, internal supporting Web Client, Draft Space and Review Space
  - 200+ SOAP/API25 calls, public surface
- **Communication-protocol changes, so can Security-protocol**
  - Enables a more contemporary security paradigm



Enablers – From API 2.5 to API 3.0

- Switch from 32-bit/x86 mode for COM+ interaction to **64-bit/x64 mode**
  - Larger memory sandbox for ever increasing data sizes, fragmentation, etc
  - BackgroundTask, TranslationOrganizer, AppPools hosting ISHCM/ISHWS/ISHSTS,…
- **Future Cost of Ownership**
  - Switching from Windows OS as host for Web/App layer into Linux OS
  - One data layer puts an Open Source or cloud optimized database layer on the horizon

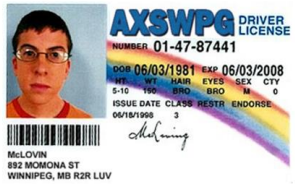
Security - Real World Scenario

1. You need resources, so off to the supermarket to buy some *good* beer, e.g.

2. The policy of the supermarket is not to sell to minors, he


photo id required

3. Your token is



4. Your token was issued before by the state, a trusted identity provider
5. After verification of your age claim, part of your token, you are authorized to buy beer

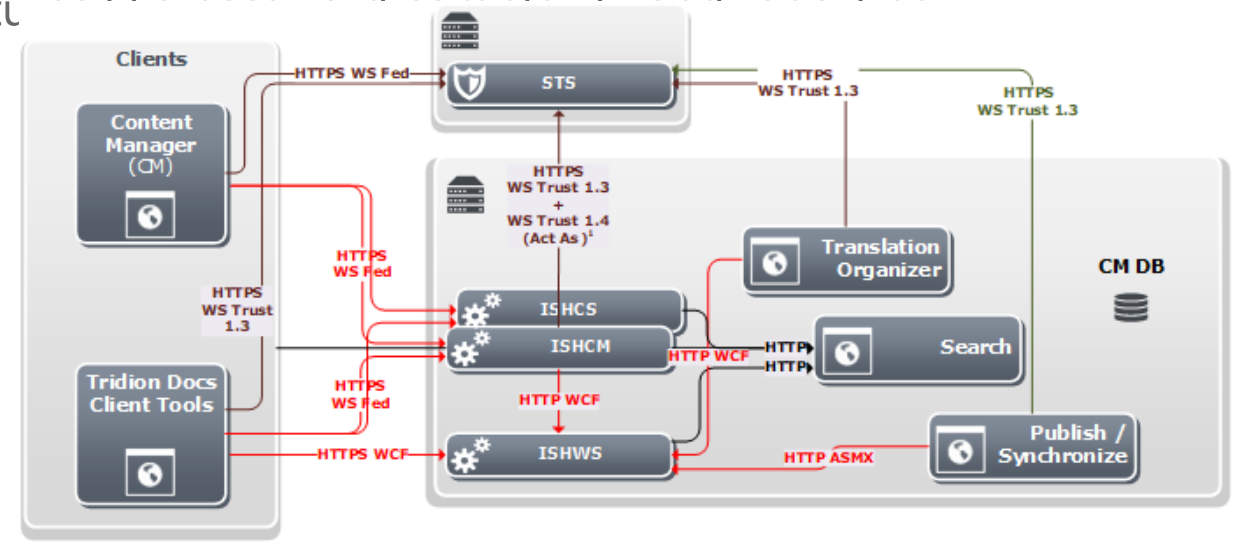
OASIS WS-* standards (2)

- WS-Federation v1.2
  - Only an OASIS standard since April 2009 and part of the larger WS-Security framework
  - Defines mechanisms for allowing disparate security realms to broker information on identities, identity attributes and authentication
  - WS-Federation builds on the WS-Trust encapsulation mechanism (the RST/RSTR) which allows protocol processing to remain agnostic of the type of token being transmitted. This enhances the interoperability and migration of customer deployed products as the industry introduces new and better security token formats.
  - WS-Federation Active Requestor Profile is a Web Services specification dealing with how applications, such as SOAP-enabled applications, make requests using these mechanisms.
  - WS-Federation Passive Requestor Profile is a Web Services specification dealing with how applications, such as web browsers, make requests using these mechanisms. In this context, the web-browser is known as a "passive requestor."

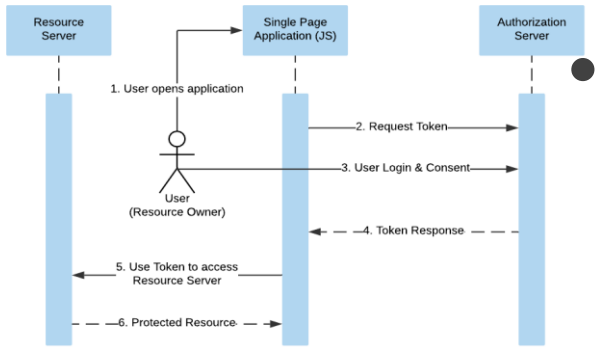
WS-Federation Protocol is a competitor of the SAML 2.0 Protocol. It is younger but backed by Microsoft and IBM. Both use SAML 2.0 Tokens! Microsoft is opening up on integrations. For example, ADFS implements the SAML 2.0 Protocol, and it is offically on the WIF backlog.

History – From OASIS WS-Federation to OpenIDConnect

- Protocols OASIS WS-Federation and WS-Trust using SAML-tokens were added in SDL LiveContent 2013/10.0.x
- Early adaptors as there were many competing standards around. **Aligned to Microsoft, IBM and OASIS.**
  - WS-Federation is diminishing over time, more web focused and **higher security demands (think 2FA)**
  - Bypassed by SAML2-P and now in turn **OpenIDConnect**
- **Built-in 'ISHSTS'**, based on IdentityServerV2, respects the same protocols to support smaller stand-alone setups
  - Never our intention to compete with Microsoft ADFS, PingIdentity, etc … especially on **Identity Provider** featu



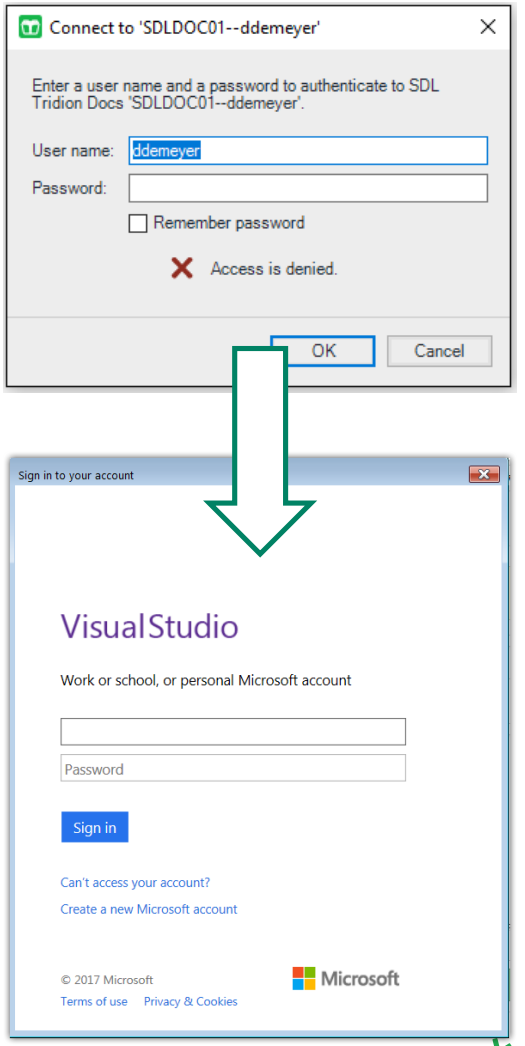
Next – From OASIS WS-Federation to OpenIDConnect

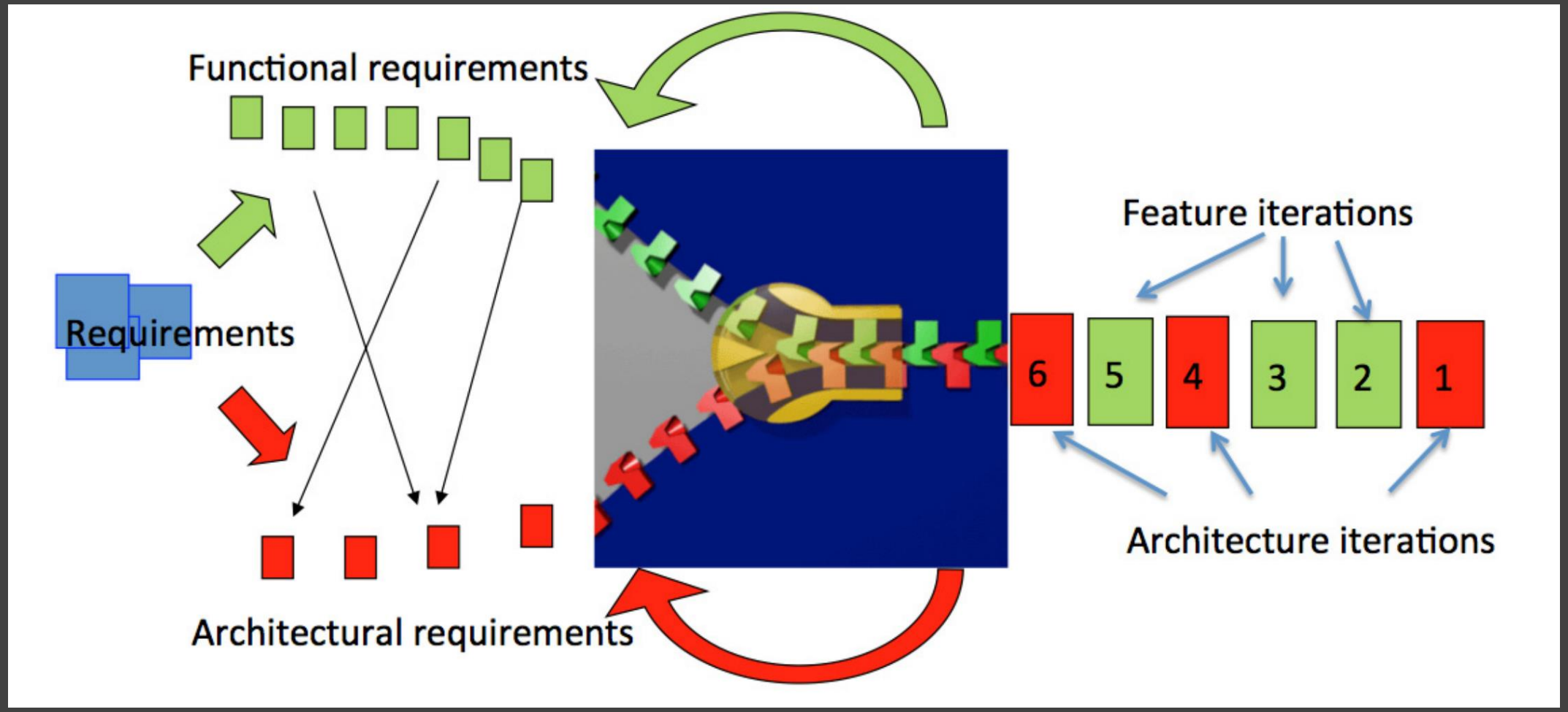


- Reuse and alignment using **Access Management**, based on IdentityServerV4, confluence with sibling Tridion Sites
  - Evolving standards, V4 dropped WS-Trust support
  - Actually .NET (Core) dropped WS-Trust (server-side)
  - No WS-Trust means ISHWS WCF-SOAP API needs change
- **Security-protocol changes to OpenIDConnect**
- Access Management aims at Federation
  - **Inside** the Tridion product we secure communication using OpenIDConnect, just like we do with WS-Fed/WS-Trust
  - **Outside** we require Federation with another Secure Token Service (STS); probably also OpenIDConnect perhaps more
  - Stepping away from CMS being an Identity Provider

Next – From OASIS WS-Federation to OpenIDConnect

- **Configures Web Services** for OpenIDConnect over Access Management (ISHAM) for
  - Draft Space & Review Space (ISHCS), Web Client (ISHCM) or Organize Space
  - Web Services (ISHWS)
- **Extend Client Tools** for browser-based 'Modern Authentication'
  - Respecting the MFA/2FA or whatever your STS enforces
  - Stepping away from 'Remember password' ownership
- **Extend server-to-server** like TranslationOrganizer, BackgroundTask or even ISHRemote need programmatic authentication
  - Commandline/Batch processing of Content Importer needs a solution
- Complete the circle to smoothen installation, upgrade, configuration, recipe, automation like ISHDeploy where possible




Functional requirements
Requirements
Architectural requirements
Feature iterations
6
5
4
3
2
1
Architecture iterations

Architectural Drivers

- Technology and Platform Support
  - COM+, then .NET Framework, now .NET-Core-on-Windows, then .NET-Core-on-Linux
  - One data layer unblocks full 64-bit, PostGreSQL, Aurora,…
  - Easier deployment over xcopy, macro-containers, micro-containers,…
- Security
  - PenTesting (XSS, CSRF, contemporary JS libraries…)
  - Next generation protocols over Access Management
- Risks & Cost
  - Upgradeability as ClassicASP mixes business logic, layout and configuration
  - People; how many know the inner-depths of COM+ server applications, or GAC
  - Build and QA process; COM+/GAC affects build servers, multi-version deployment, development (hotfixing), etc
- User Experience
  - Driven by personas
  - Localization-ready

Release Vehicles – Minor and major versions

- **It's a big task across a big, meaty product!** All these are connected, have overlap. How to approach…

- Either, adding them in a ***backward compatible*** way
  - Overlap for one release, you'll have to move anyway… Why not move during your DEV/TEST deployment phase
  - Double flavor across Client Tools, Web Services, Security-Protocol setup, Documentation, Knowledge from engineering over support/partners up to customers… Even more work, known issues, guidance as complexity and risks increase
- Either, deliver in one ***big-bang release*** covering the OpenAPI Communication protocol and Security protocol changes
  - No legacy pack, or backwards compatibility. The change is do-able in one release just like we did with Publishing or IWrite*plugin offering you massive performance improvements
  - We share your concern, as we have to rewire ourselves
  - The legacy pack is expensive and will distract us for valuable features going forward

In the pipeline enabling features for another 5+ years

- **Organize Space**, so the Web Client (ISHCM), successor built on OpenAPI
- **Web Services v3.0**, OpenAPI and Modern Authentication, successor of WCF-SOAP/WS-Trust and ASMX-SOAP/Proprietary
- Clients, like Publication Manager, rewired to browser-based **Modern Authentication** enabling MFA/2FA
- Hosted by .NET Framework and in turn **.NET** (Core)

#TXS2020

Question &s Answers

SDL

sdl.com

#TXS2020

twitter.com/SDL

facebook.com/sdlplc

linkedin.com/company/sdlplc/