Intro to the Application Lifecycle Framework (ALF) and Scenarios for the STS

Eclipse Higgins Face-to-face
Boston, July 6th – 7th 2006

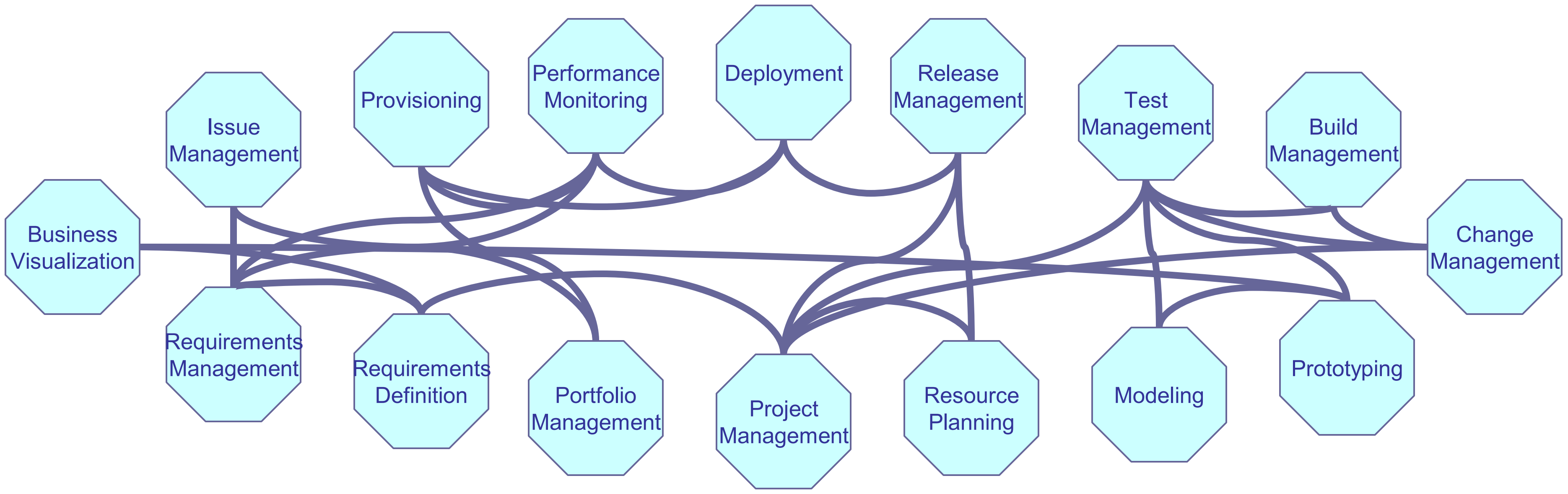
April 12th, 2006

Agenda

- Brief Introduction to ALF
- Overview of ALF Security
- Scenarios involving the STS
- Requirements for the STS
- Working together to enhance Higgins

Brief Introduction to ALF

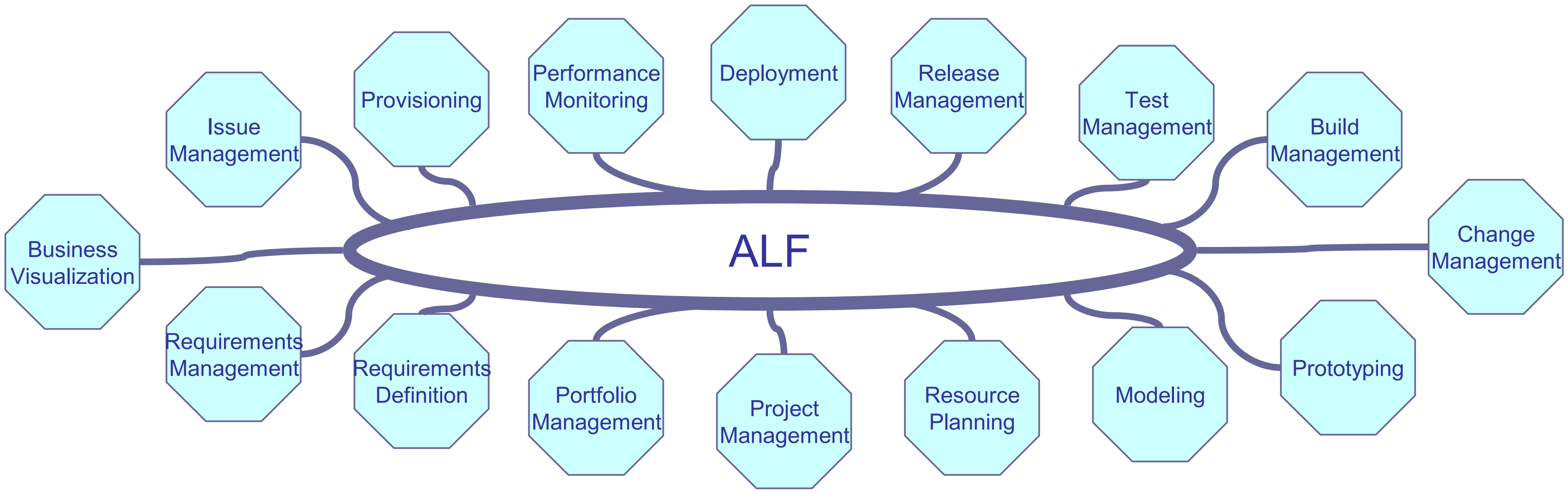
What developers are struggling to support



Point-to-point integration of n tools can mean

up to n(n-1)/2 combinations; This does not scale!

A better approach – Model: electric wall plug



Integration of n tools with ALF requires n integrations; linear growth of connections

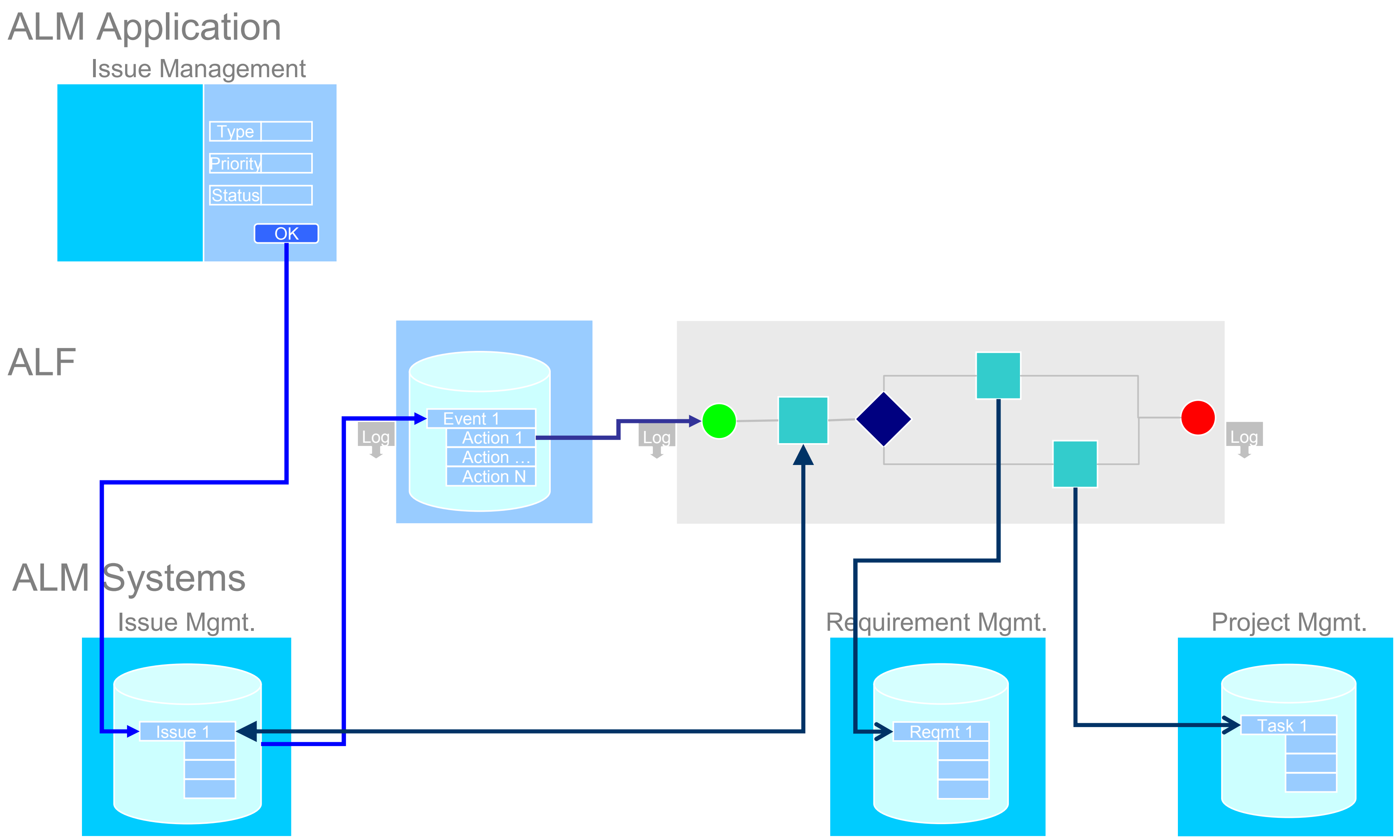
Project Objectives

1. Provide a SOA-based interoperability framework infrastructure for cross-tool ALM solutions
2. Leverage Eclipse, open source components, and industry standards
   - WS-*, BPEL, SAML, WS-I, …
   - Apache Axis
3. Develop common & extensible domain-specific vocabularies for improved interoperability
4. Provide conformance rules for varying levels of participation

Major Components of ALF

- Event Manager
  - The first ALF Contribution
- *BPEL Engine*
  - *Pluggable – supply your favorite engine*
  - *Currently using ActiveEndpoints*
- *Security Token Server*
  - *Use and extend the IBM STS contribution to Higgins*
- Administration tools
  - Eclipse Plug-ins
  - Event-Action Mapper undergoing Eclipse IP review
  - *Pluggable BPEL designer - Currently using ORACLE BPEL Designer for demos*
- Working Samples of ALF in action
  - Use web service stubs to avoid licensing issues with commercial tools
- Vocabularies
  - Exemplar use cases, data models, XML Schemas and WSDL for tool domains
- Processes
  - Process exemplars

ALF Use Case


ALM Application
Issue Management
Type
Priority
Status
OK
ALF
Event 1
Action 1
Action …
Action N
Log
Log
Log
ALM Systems
Issue Mgmt.
Issue 1
Requirement Mgmt.
Reqmt 1
Project Mgmt.
Task 1

ALF Timeline

- Proposed in April 2005
- Creation Review in July 2005
- Proof-of-concept (POC) demo in Feb 2006
    - Extended to EclipseCon in March 2006
- Currently working on:
    - Downloadable sample applications
    - Vocabularies
    - Security and SSO
        - Traditional, user-oriented
        - Conveying security context through BPEL process
- Goal: 1.0 Release Candidate by early November 2006

Overview of ALF Security

What is the focus of ALF Security

- Initial focus (for RC1) is on Authentication
  - Authentication of users of web browser and desktop tools
    - Using WS-Trust & WS-Federation Passive Requestor Profile
    - SAML token (ALF TGT)
  - Conveying credentials to all the programs invoked via web services by a ServiceFlow (i.e., BPEL process)
    - Using WS-Trust & WS-Federation Active Requestor Profile
    - SAML Token (ALF TGT and ALF ST)
- Later phase focus Expands Authentication and add Authorization
  - Authentication of users of desktop and plug-in-based tools
    - Likely to leverage Corona and Eclipse platform OGSi security initiatives
    - Likely to use JAAS (we may accelerate if possible)
  - Optional and/or later focus is on Authorization at the admin and serviceFlow and perhaps tool level
    - Not addressing privileges within tools
- Once infrastructure is in place, add options for message confidentiality and integrity

Key Standards ALF is based on

- Standards for ALF RC 1 (Oct 2006)
  - WS-Security
    - UserNameToken
    - SAML Assertion
  - WS-Trust
  - WS-Federation
    - For signoff
    - Active Requestor Profile (Web services)
    - Passive Requestor Profile (Web application)
  - SAML Assertion (1.1 and 2.0)
  - WS-Policy and WS-SecurityPolicy (Static administration for RC 1.0)
- Standards for post ALF 1.0
  - WS-Security BinarySecurityToken
    - For credentials in form of Kerberos and x.509 certificates
  - SAML Protocol (as alternative to WS-Trust)
  - Dynamic discovery and exchange (per WS-Trust)

Annoying Realities

- Most tools today do not trust an external identity to perform authentication
    - The approach suggested by ALF “factors out” authentication functions from the tools
- Tools do not use unified identifiers for users
- Most shops have created some custom mechanism that partially addresses the SSO requirements

Scenarios involving the STS

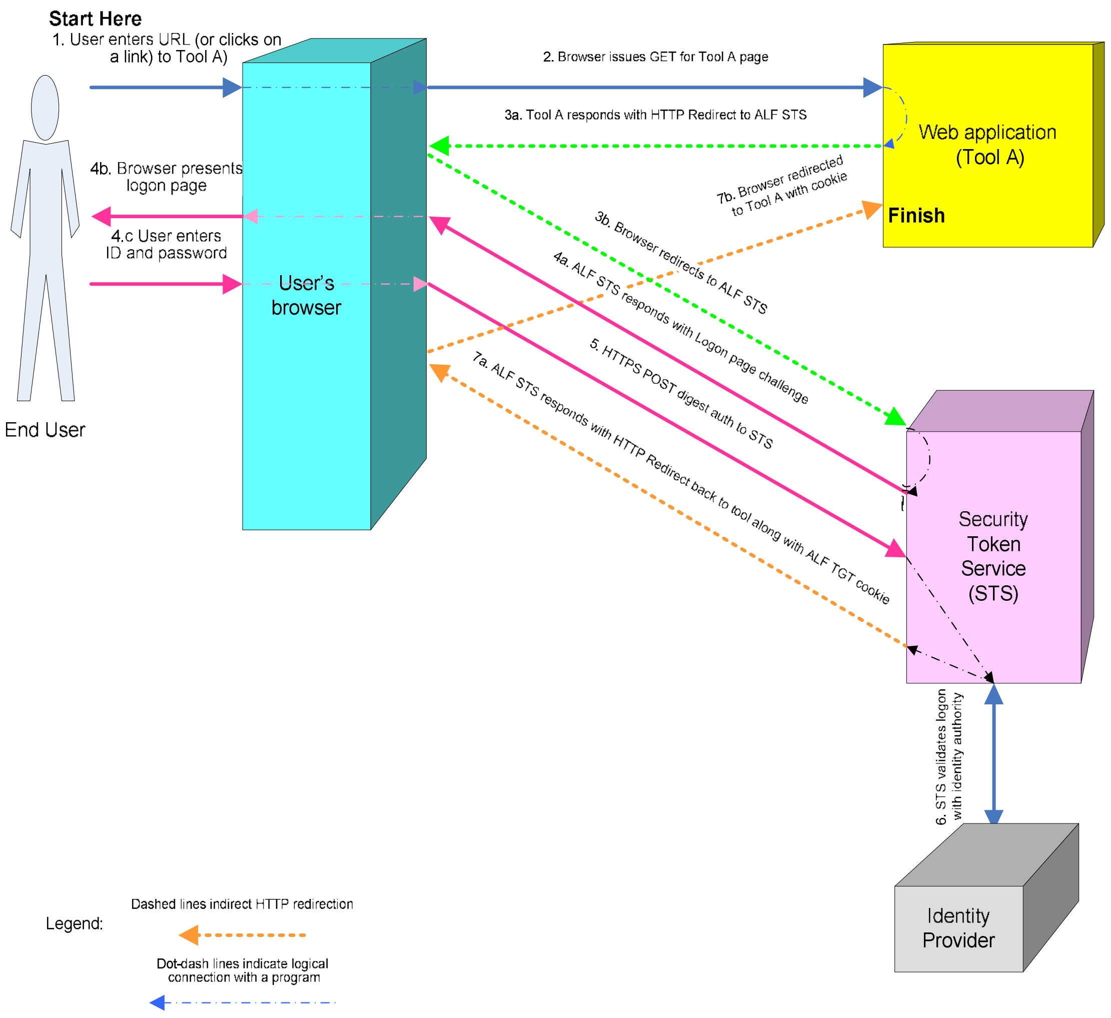
Scenario 1 – Traditional SSO

- Starting with web-browser-based tools
- User authenticates to first tool
  - Obtains a token (ALF TGT) with lifetime of typical workday
  - Token communicated via Passive Requestor Profile (QueryString, POST, RST & RSTR) or traditional cookies
  - Token accepted by other tools avoids user re-authenticating
- Aspects
  - Keystore for cross-domain tools
    - [Keystore for SAML tokens??]
  - ALF admins at installation sites determine tradeoffs (convenience, vs corporate policy, …)
  - Extensions to:
    - Eclipse-based tools (may wait for Equinox)
    - Desktop tools (initially via tool logon, eventually via OS logon)

Scenario 1 - Specific use cases

- Initial issuance of a token
- Subsequent validation of token as other tools are presented with the token
  - Tools can validate the signature in the token, but cannot determine whether token has been revoked
- Token revocation (user signoff)

Scenario 1 – Traditional SSO


Start Here
1. User enters URL (or clicks on a link) to Tool A)
2. Browser issues GET for Tool A page
3a. Tool A responds with HTTP Redirect to ALF STS
Web application (Tool A)
Finish
7b. Browser redirected to Tool A with cookie
4b. Browser presents logon page
4.c User enters ID and password
User's browser
3b. Browser redirects to ALF STS
4a. ALF STS responds with Logon page challenge
5. HTTPS POST digest auth to STS
7a. ALF STS responds with HTTP Redirect back to tool along with ALF TGT cookie
End User
Security Token Service (STS)
6. STS validates logon with identity authority
Identity Provider
Legend:
Dashed lines indirect HTTP redirection
Dot-dash lines indicate logical connection with a program

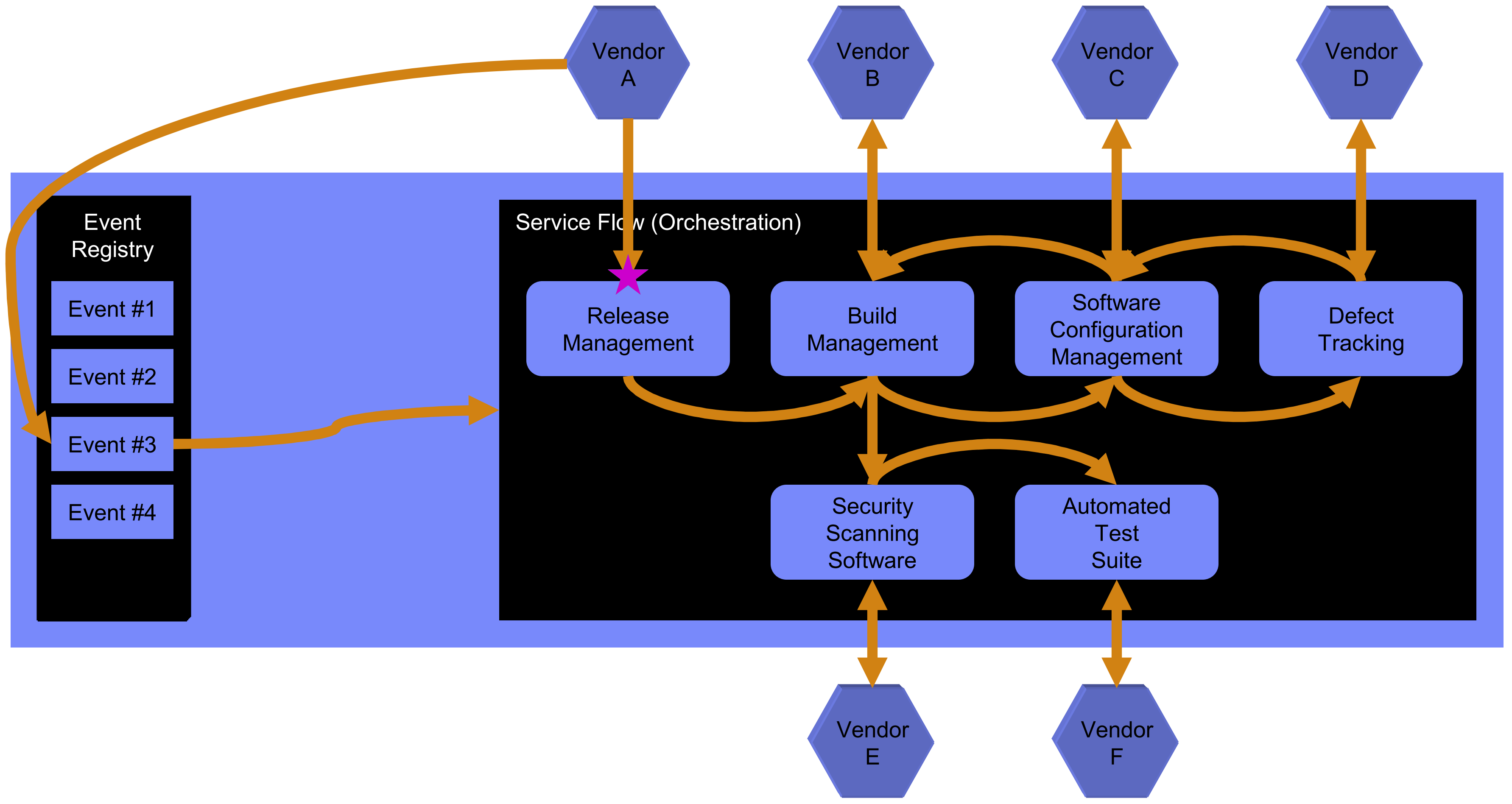
Scenario 2 – Conveying a security context via BPEL

- ALF ServiceFlow Token (ALF ST) has a longer time-to-live
- Is bound to the BPEL process via an EventID

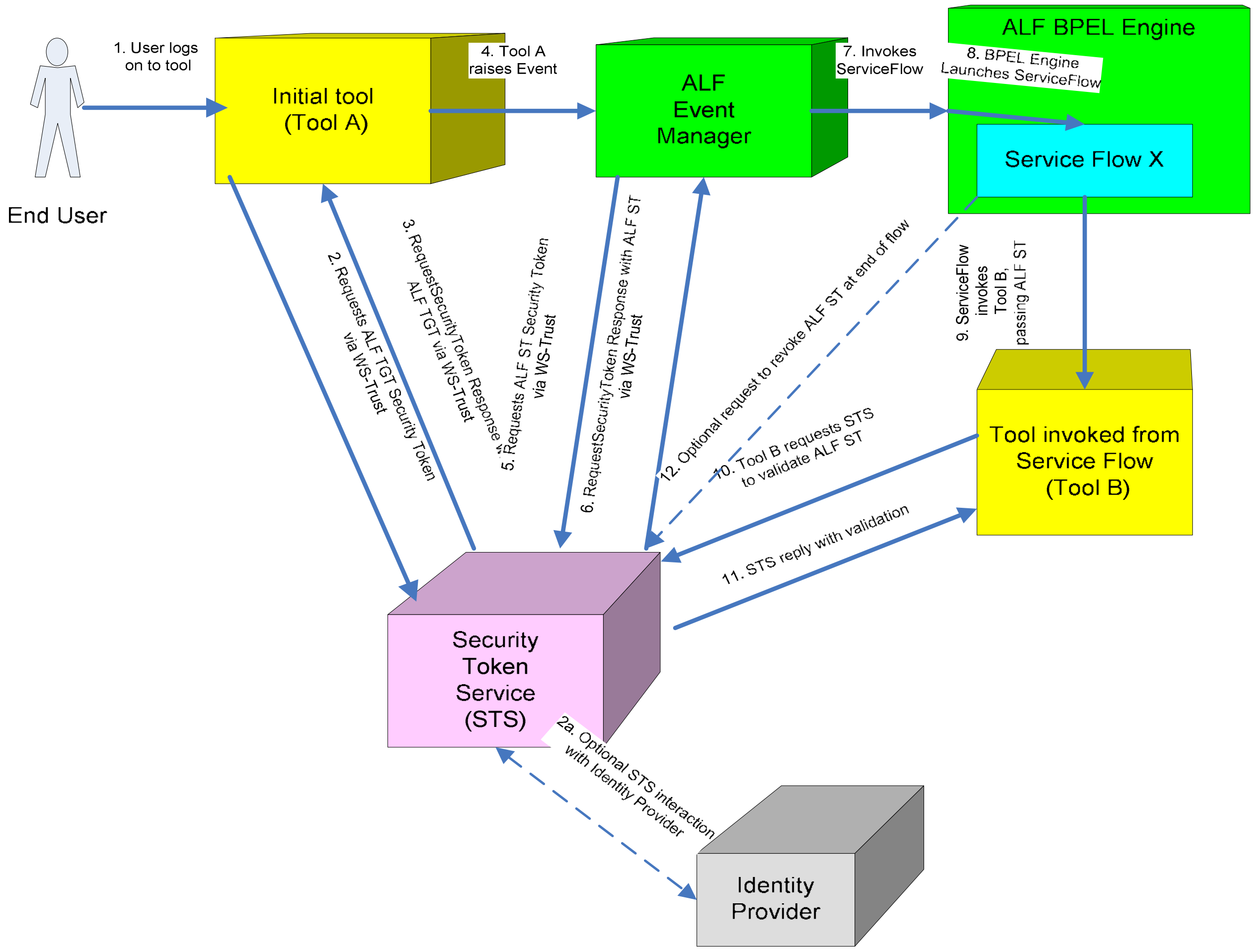
- Note: Ideally extensions to the BPEL engine would handle conveying the security context, but the BPEL engine is pluggable
  - So:
    - EventManager submits an ALF TGT to obtain an ALF ST

Scenario 2 example

How it all works: Example Service Flow Orchestration



ALF SSO Scenario


End User
1. User logs on to tool
Initial tool (Tool A)
4. Tool A raises Event
ALF Event Manager
7. Invokes ServiceFlow
ALF BPEL Engine
8. BPEL Engine Launches ServiceFlow
Service Flow X
9. ServiceFlow invokes Tool B, passing ALF ST
Tool invoked from Service Flow (Tool B)
2. Requests ALF TGT Security Token via WS-Trust
3. RequestSecurityToken Response with ALF TGT via WS-Trust
5. Requests ALF ST Security Token via WS-Trust
6. RequestSecurityToken Response with ALF ST via WS-Trust
12. Optional request to revoke ALF ST at end of flow
10. Tool B requests STS to validate ALF ST
11. STS reply with validation
Security Token Service (STS)
2a. Optional STS interaction with Identity Provider
Identity Provider

Scenario 2 - Specific use cases

- Initial issuance of a token for a BPEL process instance
  - EventManager presents an ALF TGT and EventID to obtain an ALF ST
- Subsequent validation of token as other tools are presented with the token
  - Tools can validate the token’s integrity via its signature, but cannot determine whether token has been revoked
- Token revocation (service flow terminated)

Scenario 3 (a variation suggesting a headless ISS?)

- How to convey the appropriate security context to a tool that does not use the standard corporate login (e.g., not LDAP directory based)
  - For example, Z/OS tools
  - For example, custom tools that still use a proprietary database of users
- Provide the option to use a generic, “System ID”, if needed
- Can a variation on the Higgins context and Identity Selector notions be use to select the right credentials to present to such a tool?
- And how would this work with the BPEL process and ServiceFlow token?
  - Would need a headless identity selector that selected the appropriate identity based on policy

Requirements for the STS

Requirements for the STS (1 of 2)

- Ability to extend the SAML Token
  - ALF TGT
  - ALF ServiceFlow Token
- Ability to authenticate against commonly used identity stores
  - LDAP, AD, ADFS, custom database
  - STS should use pluggable model
- Ability for STS to remember which tokens it has issued and remember revocations
- Ability to obtain a token, given credentials:
  - UsernameToken (initial requirement)
  - Kerberos (as you would obtain from an operating system logon)
  - X.509 certificates (primarily to support smart cards)

Requirements for the STS (2 of 2)

- Credentials mapping – (ISS?) presenting the appropriate credentials for a tool
  - Which ID to use should be based on policy
- Option to use static WS-Policy files rather than dynamically obtaining policy via Metadata Exchange

Working together on Higgins

How to work together

- ALF would like to drop plans to build an STS and focus on ensuring that the IBM STS contribution to Higgins meets ALF’s requirements
- Suggest that ALF become:
  - Initially an early adopter, consumer, tester
  - A contributor of enhancements, fixes
    - E.g. Passive Requestor Profile servlet front end
  - Possibly eventually a Higgins committer
- After ALF 1.0 RC, start to incorporate other aspects of Higgins
  - E.g. the Identity Selector
    - Perhaps a headless, server-side variant for mapping credentials that are presented to tools

What is ALF building and can contribute to Higgins

Aids for tools to enable to SSO

- Library of helper functions
  - For Java-based clients and server-based tools
- Possibly a Web service gateway for tools that don’t support WS-Security
  - Intercepts messages, strips off and handles security headers
  - The will pass on web service messages along with logon/logoff messages
  - Adds security headers back on to outgoing messages