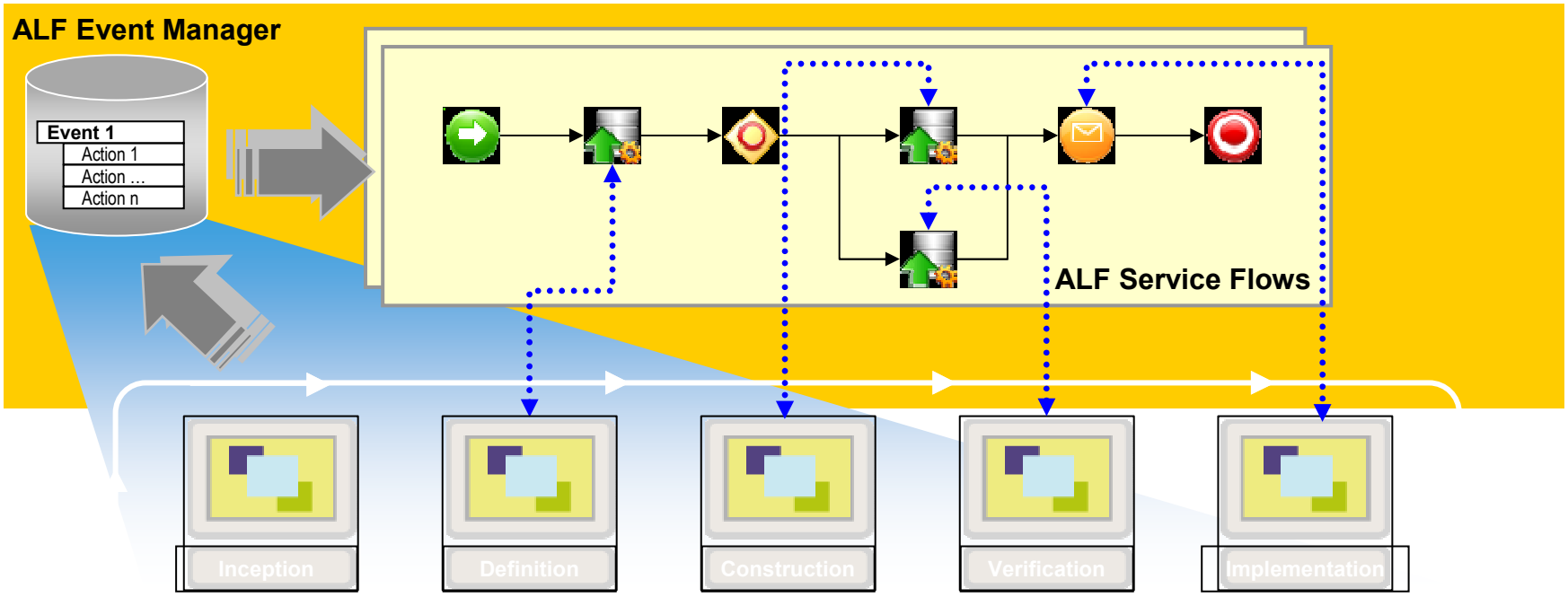
ALF SSO – ALF Federation Server For Higgins
Multiprotocol Server meeting, Dec 17th 2007

A quick introduction to ALF SSO

What is ALF?

- The Eclipse Application Lifecycle Framework (ALF) Project is an open source project under the Eclipse Foundation that provides a framework for tool integration and interoperability
- Based on SOA and Security standards
- ALF inverts the usual problem of tool integrations by putting the customer in charge, not the vendors
  - Integrations are implemented using standard languages (BPEL and WSDL)
  - The underlying framework is open source
  - If the integration breaks when you upgrade one or the tools, the customer can fix it

ALF: Is About Process Integration



What is ALF SSO

- Once you have the Event and ServiceFlow processing working, you need to know who was responsible for initiating all the activity performed in a ServiceFlow
  - Tools tend to have authorization mechanisms to control who can perform what operations on what data
- Web service standards provide the mechanism for conveying an identity
  - WS-Security defines a standard way to convey an identity through web service calls
  - SAML defines a standard token format
  - *But where do we get the identity in the first place?*

ALF Single Sign On

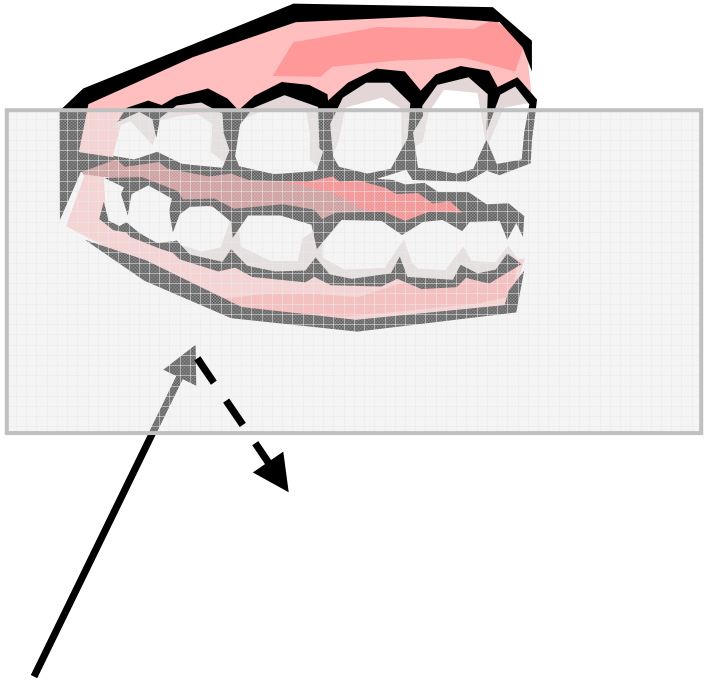
- ALF has two related mechanisms:
  - A way for a user at a tool to obtain a standardized token that proves he has authenticated
    - Similar to obtaining a drivers license at the DMV
    - Token life is typically set to approximate a workday
  - A means for conveying that standardized token through Events and the tools that are invoked by the resulting ServiceFlows
    - But ServiceFlows may run for longer that a workday
    - So a variant with longer life that is bound to the service flow is used

Concepts and Trends

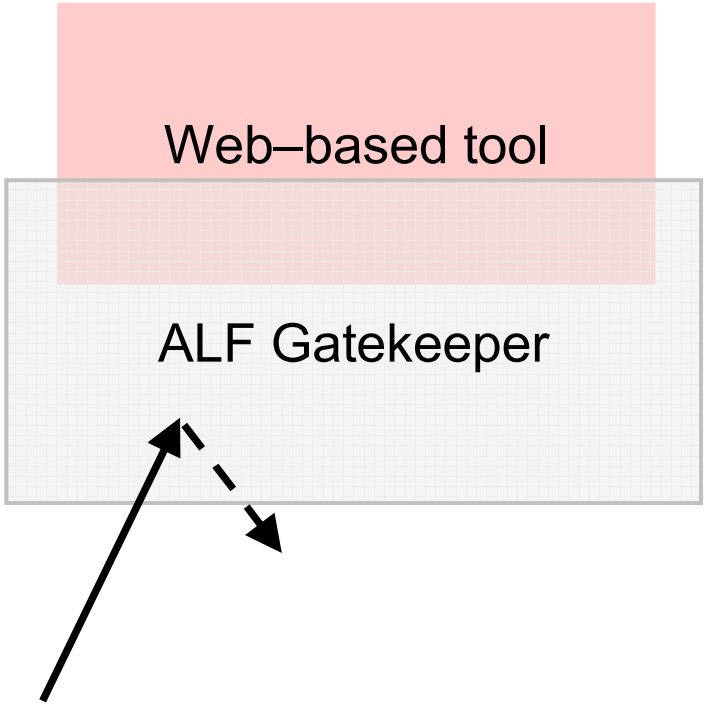
Security Tokens = Drivers License

- A Security Token is a collection of claims made by an authority about a subject
  - Similar to your drivers license
- To obtain a drivers license, you go down to the DMV, present some proof of qualification (e.g. test score and credentials issued by other authorities) and the DMV issues a drivers license that you can use to convince other parties
  - You are who you claim to be, and
  - An act of authentication (verifying your credentials) had occurred
- Obtaining and using a Security Token:
  - applications (on behalf of users) present a user's credentials to a Security Token Server (STS) operated by an authority (the "DMV")
  - the STS checks the credentials and issues a token (a SAML Assertion in this case) that your application can present to other applications
  - applications trust the claims in Security Tokens because they are signed by the trusted authority

Gatekeeper – “Invisible protective shield”



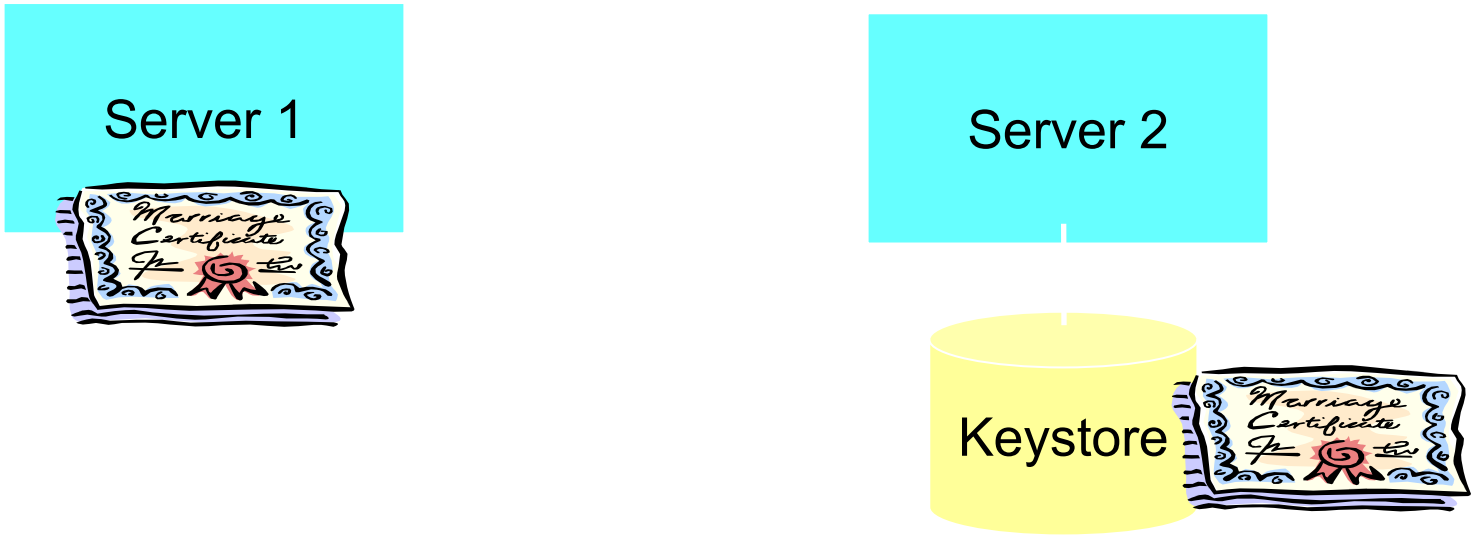
Unwanted germs



Messages without proper user credentials

Trust – Leveraging PKI

- A Trust Relationship is established between servers by placing a copy of Server 1's public key on Server 2
- Server 2 can then verify that messages signed by Server 1 (using Server1's private key) were indeed issued by Server 1
  - Because of the complementary relationship between the public and private keys of a PKI key pair, Server 1's public key is the only key that can decrypt the information that was encrypted the corresponding private key
  - So Server 2 can verify that messages supposedly from Server 1 actually cam from Server 1 because only Server 1 would have that Private key



Trends in Security

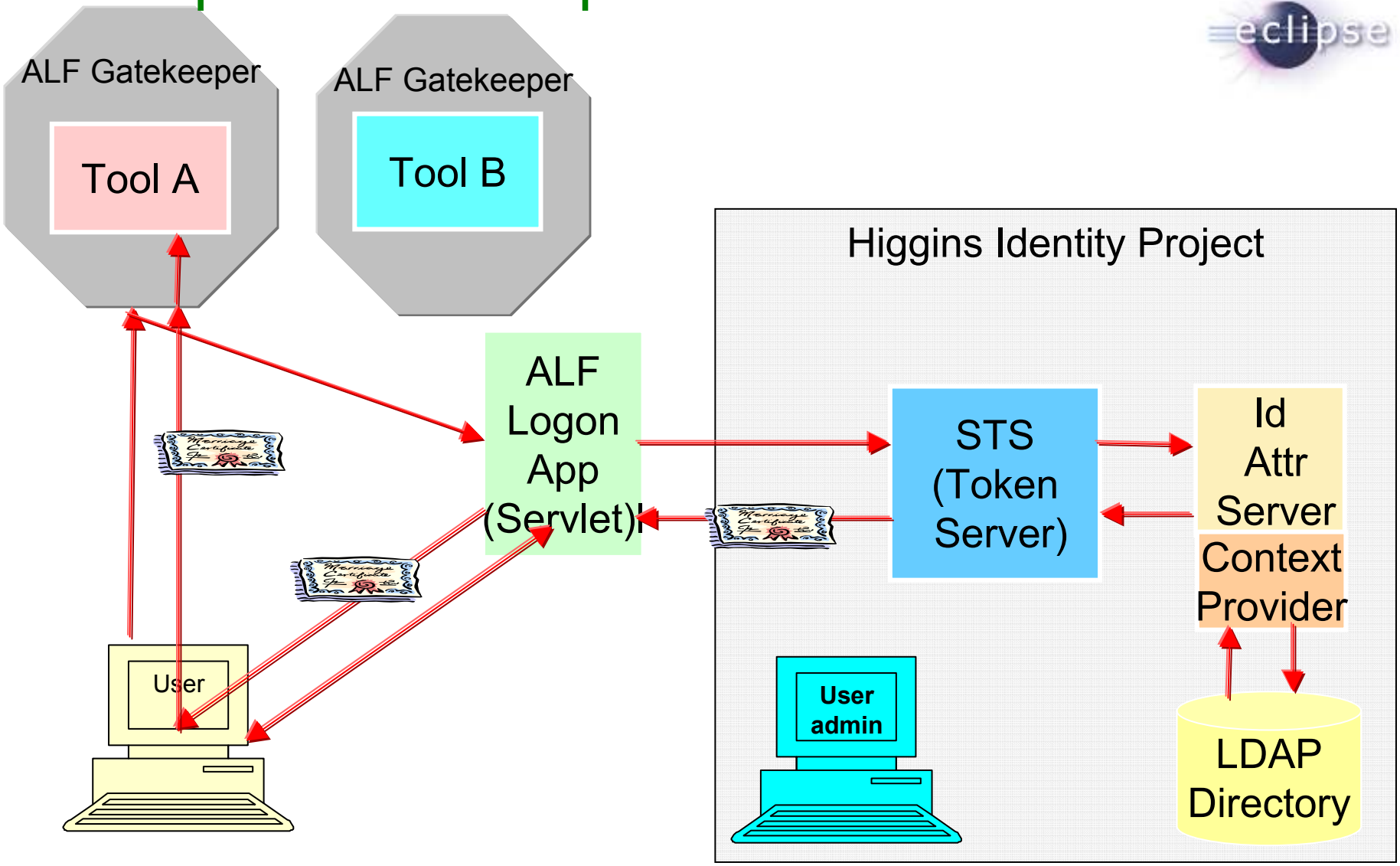
- "Factoring out" authentication from the application
  - Avoids having your critical security logic written by "Joe Developer"
  - In production, you can focus more infrastructure hardening efforts on just a handful of security components
- Make it easier for tools and applications to implement security
- Adoption of SAML Assertion as a token format
- Avoid passing around the password

ALF SSO: Standards, Components, & Technology

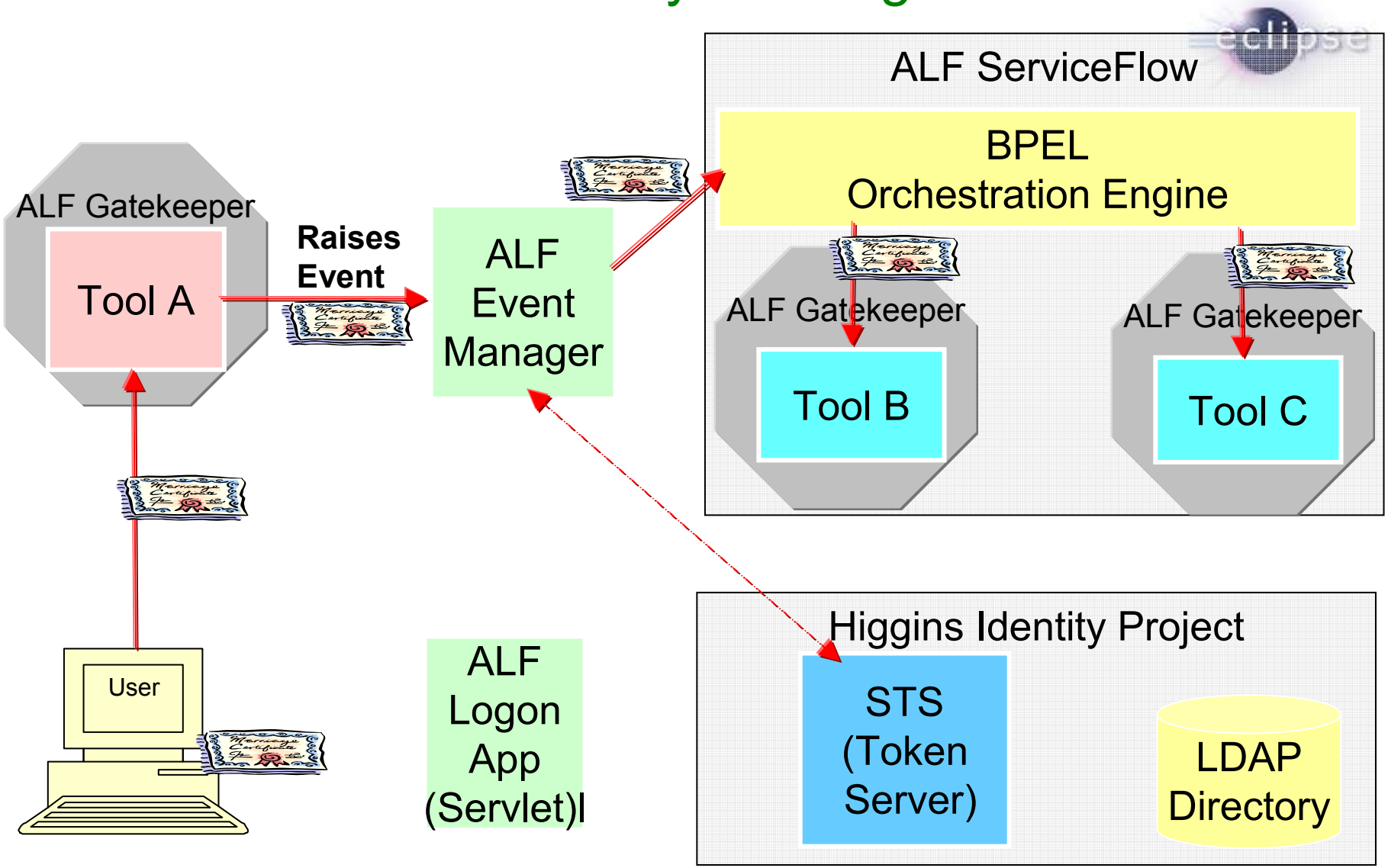
Key Standards used

- Both ALF and Higgins use
  - WS-Security
    - Username Token
    - SAML Token
  - WS-Trust
  - WS-Federation
    - Passive Requestor Profile (Web application)
  - SAML Assertion 1.1
  - XML Canonicalization and Digital Signature
- Higgins also uses
  - LDAP

Sample SSO components and flow



ALF Orchestration Identity Passing

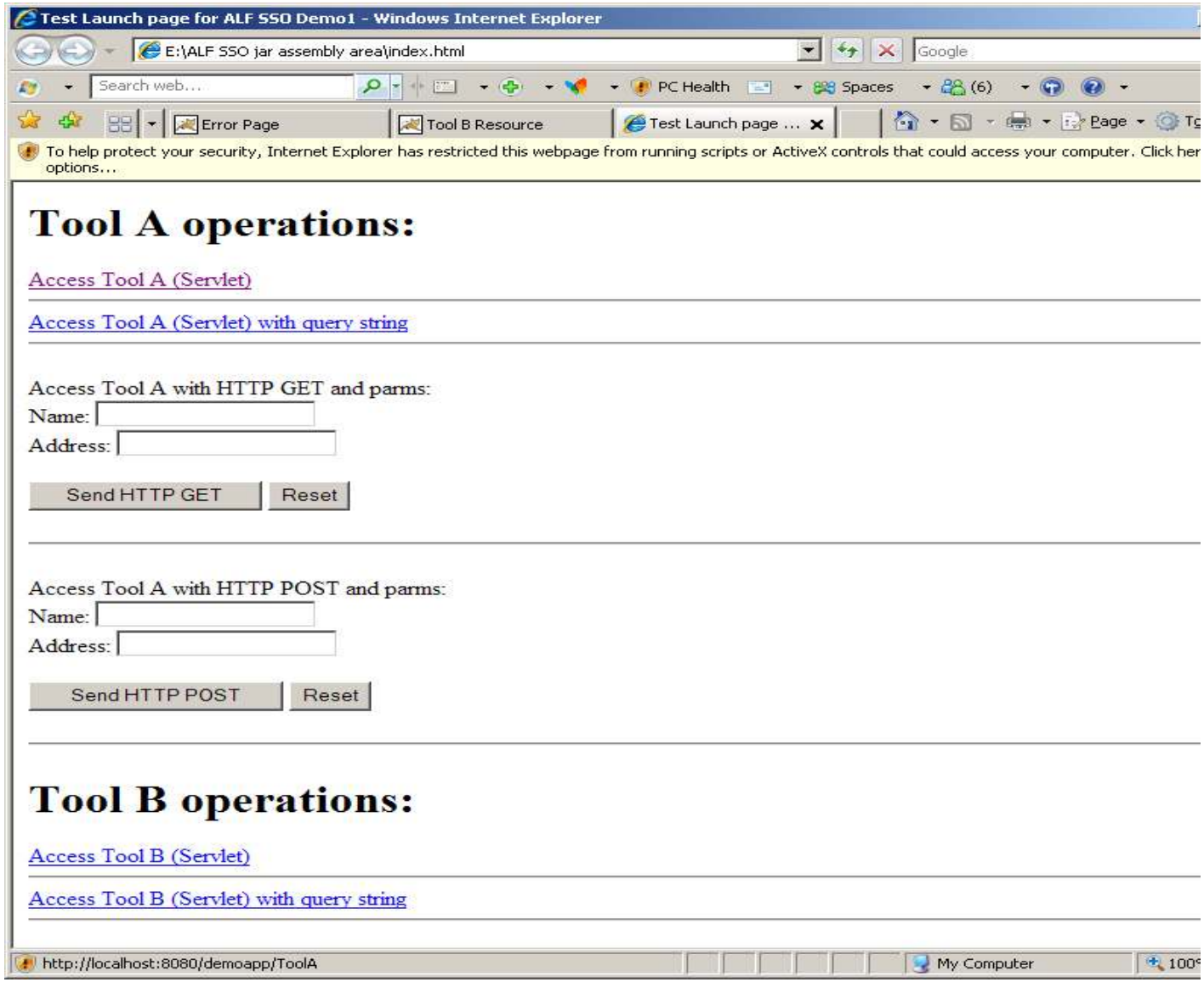


Components

- ALF:
  - **Gatekeeper** – A Servlet Filter – rejects incoming messages that lack the proper credentials
    - A filter listening on all ports on localhost.
  - **Logon Servlet** – Interacts with end user to obtain credentials
    - Http://localhost:8080/ssologon or Https://localhost/ssologon
- Higgins:
  - **Security Token Server** – Servlet (Axis generated w/WST)
    - Http://localhost:8080/TokenServer
  - **Aldentity Attribute Server** (IdAS) – Abstraction layer to present attributes from authentication authorities.
    - A .jar included in the SecurityToken Server servlet
  - **Context Provider** – An adapter to back end authentication authorities, such as an LDAP directory
    - A .jar included in the Security Token Server servlet
- Miscellaneous parts:
  - Demo launch page – A simple HTML page or HTML link to Tool A and ToolB
  - Tool A and Tool B – 2 simple servlets to represent web-based tools
    - Servlets hosted at http://localhost:8080/demoapp/ToolA
  - An LDAP-enabled Directory Server
    - Microsoft ADAM, LDAP://localhost:389

ALF SSO: Traditional web-based user authentication

At a browser, using a link or typing a URL to access a resource at ToolA: http://localhost:8080/demoapp/ToolA

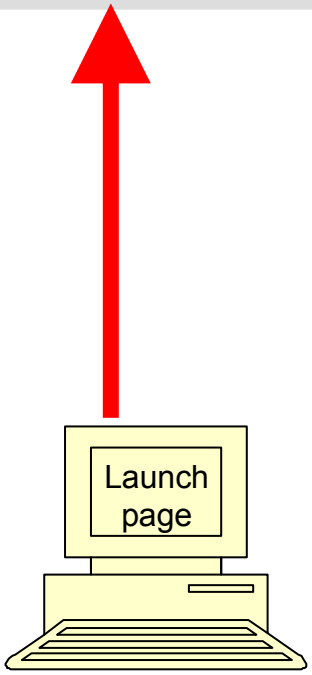


1. User at browser attempts to access Tool A

Tool A

Tool B

Gatekeeper (filter)



Logon
Appl

STS
(Token
Server)

Id
Attr
Server

Context
Provider

LDAP
Directory

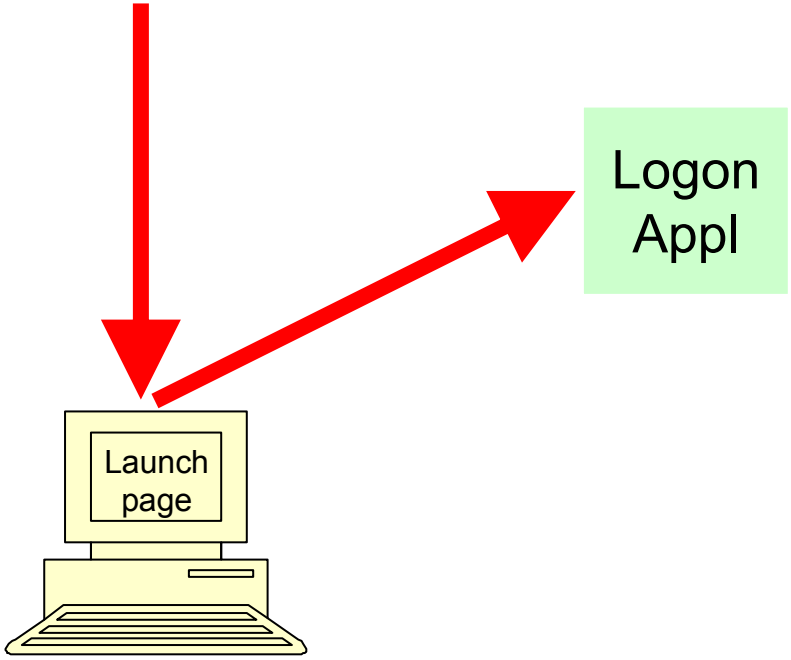
2. Gatekeeper finds no token, so redirects

Tool A

Tool B

HTTP Redirect per the WS-Federation Passive Requestor Profile

Gatekeeper (filter)



Logon Appl

STS (Token Server)

Id Attr Server

Context Provider

LDAP Directory

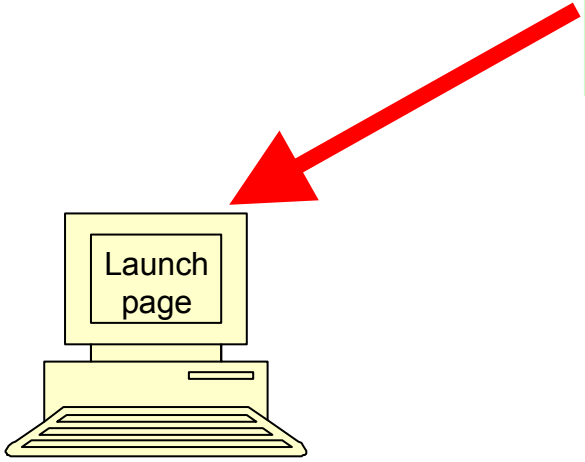
3. Logon App presents a logon page to obtain credentials from the user

Tool A

Tool B

Gatekeeper (filter)

Logon Appl



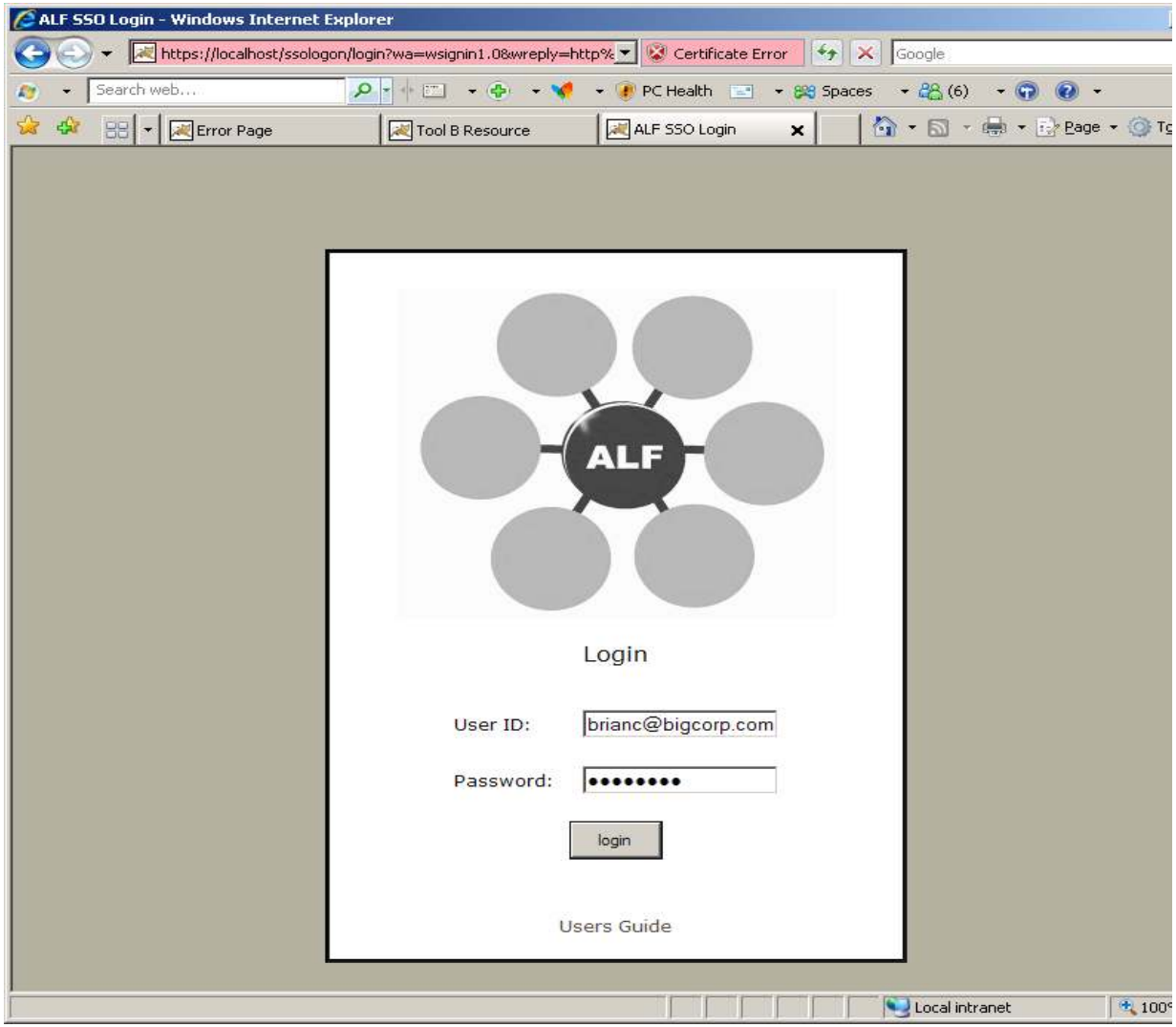
STS (Token Server)

Id Attr Server

Context Provider

LDAP Directory

The user fills in the logon page



4. When the user clicks, the credentials are sent to the Logon Appl over SSL

Tool A

Tool B

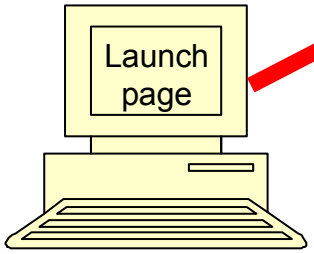
Gatekeeper (filter)

Logon Appl

STS (Token Server)

Id Attr Server

Context Provider



LDAP Directory

5. Logon App issues RST to STS

Tool A

Tool B

WS-Trust
Request Security
Token (RST)

Gatekeeper (filter)

Logon
Appl

STS
(Token
Server)

Id
Attr
Server

Context
Provider

LDAP
Directory

6. STS calls IdAS to authenticate

Tool A

Tool B

Gatekeeper (filter)

Logon Appl

STS (Token Server)

Id Attr Server

Context Provider

LDAP Directory

7. Context Provider accesses LDAP Directory

Tool A

Tool B

Gatekeeper (filter)

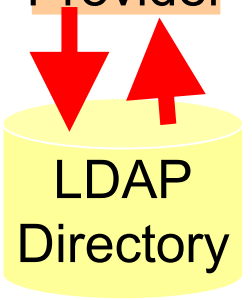
Logon Appl

STS (Token Server)

Id Attr Server

Context Provider

Maps Claim names to Attribute names in underlying store



8. IdAS returns verification to STS

Tool A

Tool B

Gatekeeper (filter)

Logon Appl

STS (Token Server)

Id Attr Server

Context Provider

LDAP Directory

9. STS returns response to Logon (RSTR)

Tool A

Tool B

WS-Trust
Request Security Token
Response (RSTR) which
contains the SAML token

Gatekeeper (filter)

Logon
Appl

STS
(Token
Server)

Id
Attr
Server

Context
Provider

LDAP
Directory

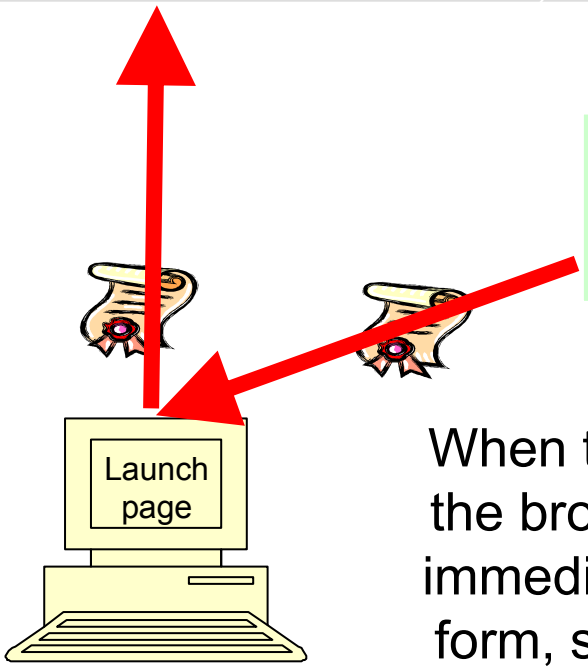
10. Logon App “redirects” request, now with token, back to Tool A

Tool A

Tool B

Redirect via a Forms POST per the WS-Federation Passive Requestor Profile

Gatekeeper (filter)



Logon Appl

STS (Token Server)

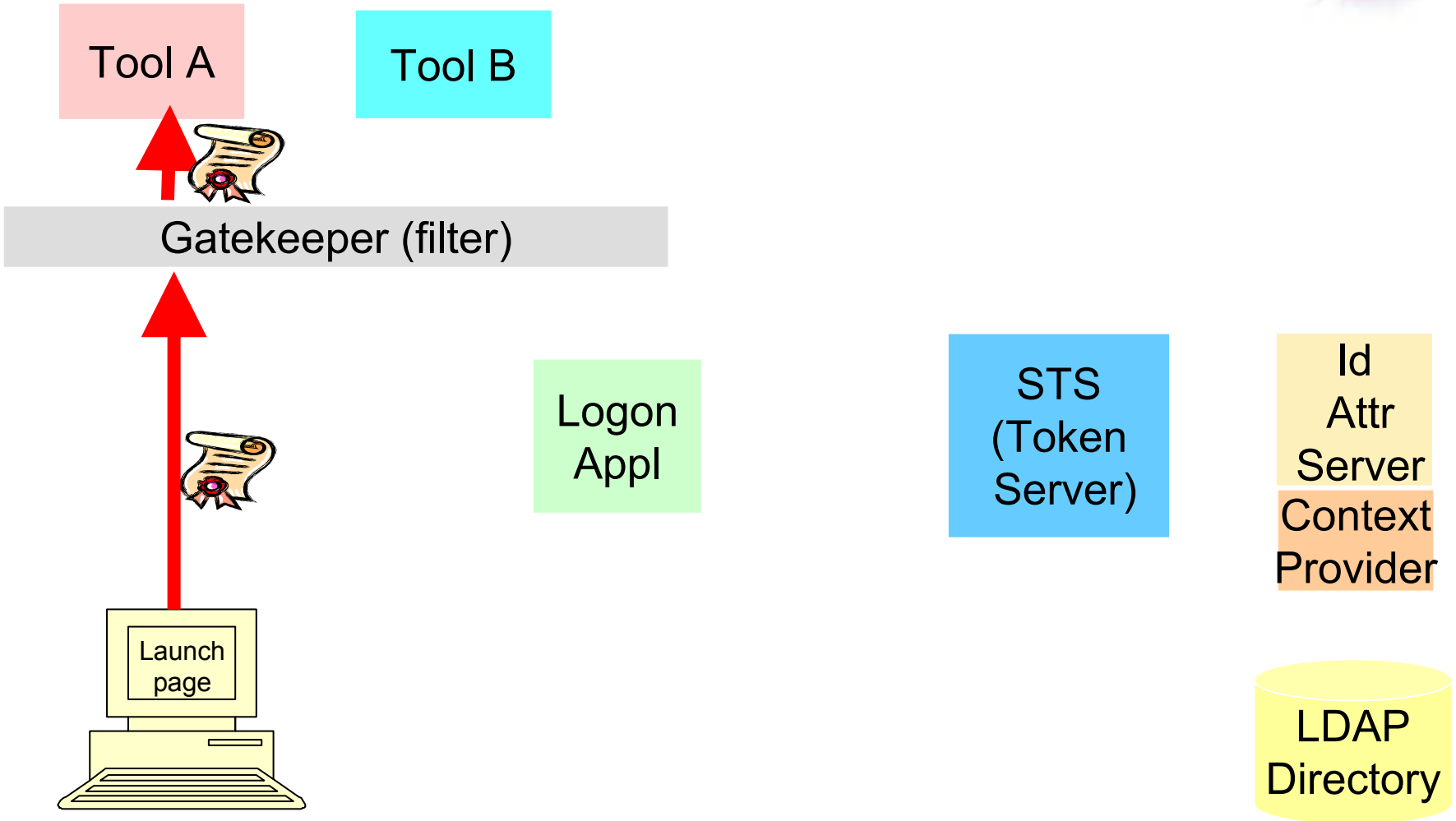
Id Attr Server

Context Provider

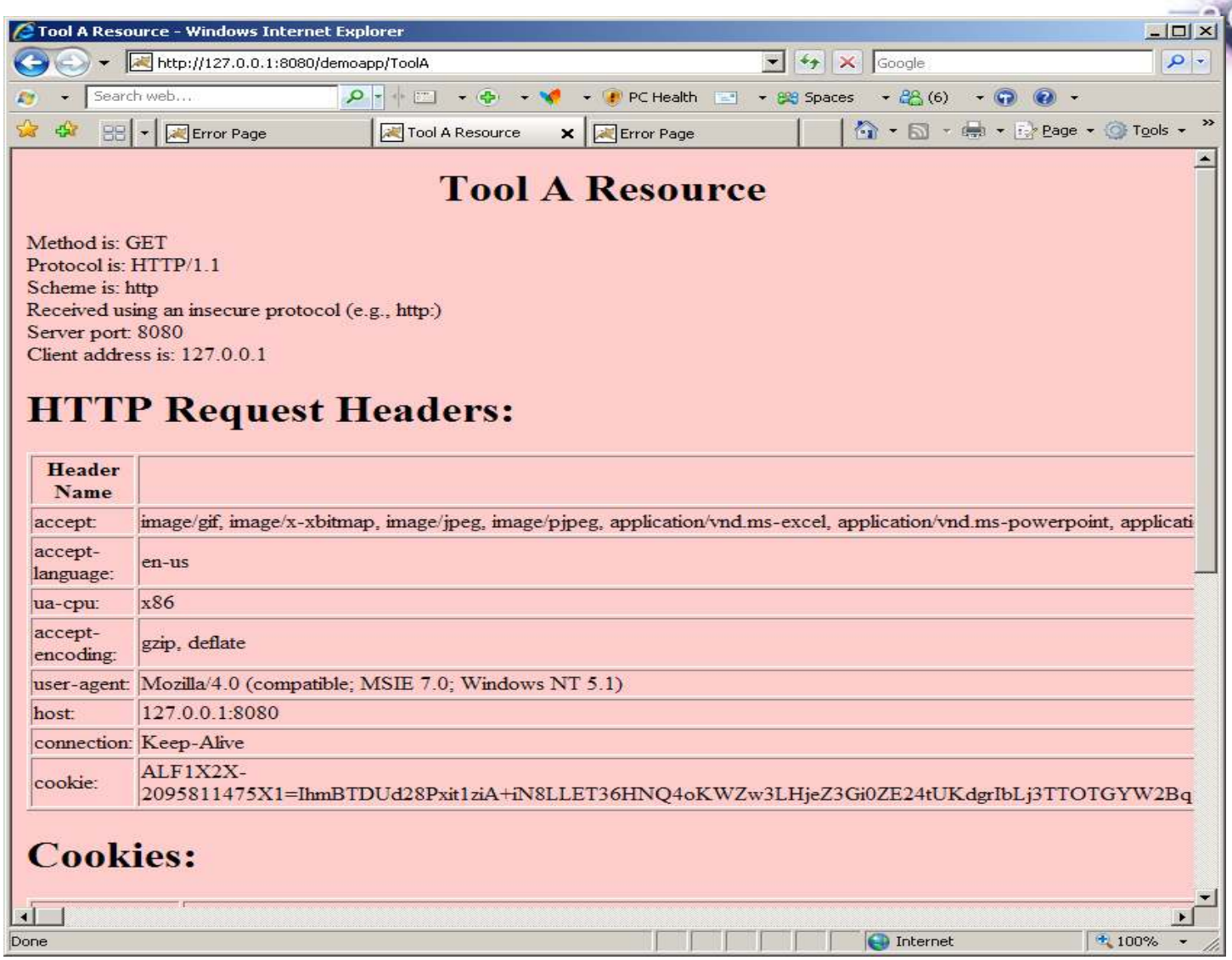
LDAP Directory

When the page loads in the browser, JavaScript immediately submits the form, so the user never sees this step

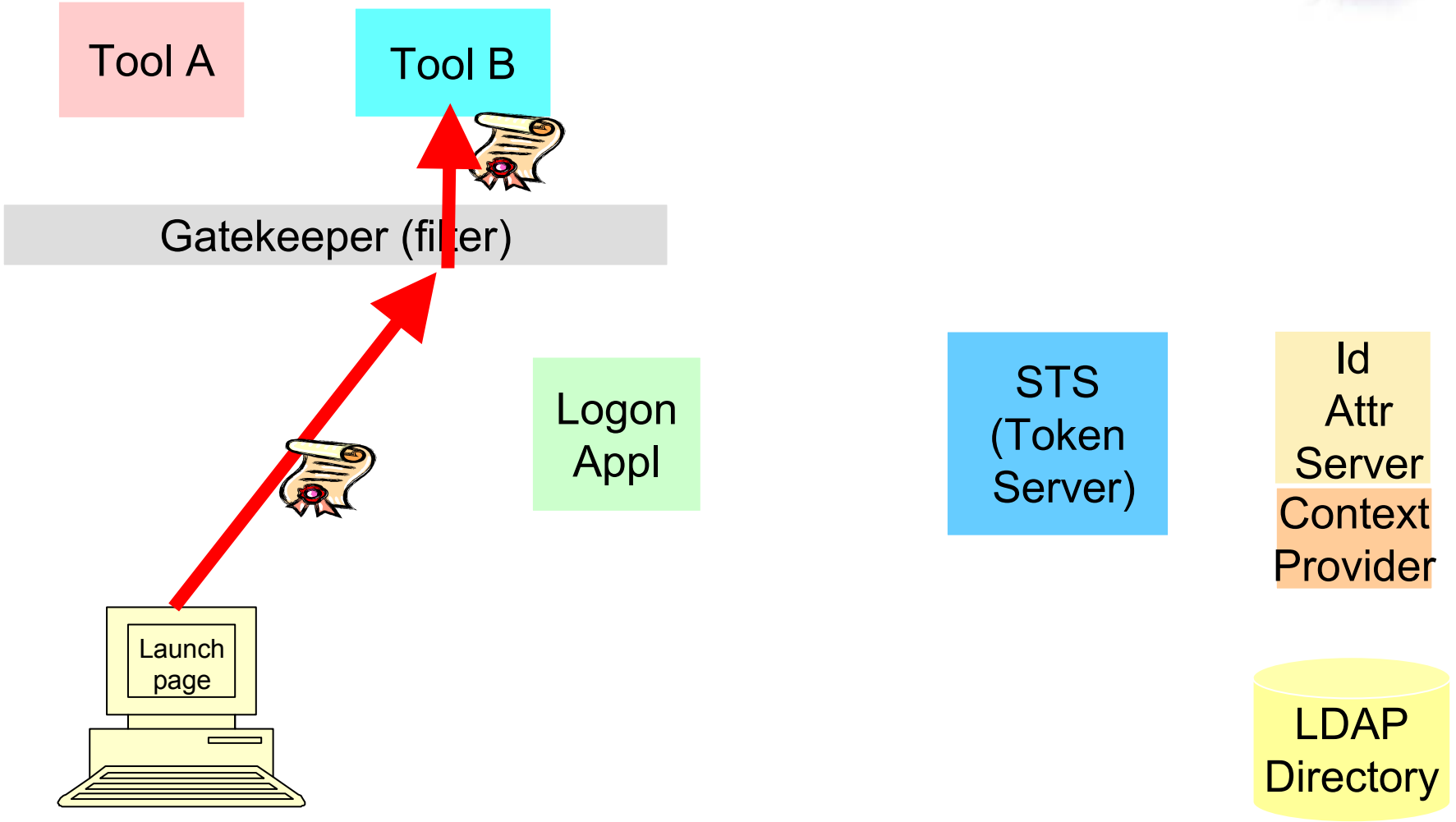
11. Gatekeeper accepts request because it contains the token and lets it through to Tool A



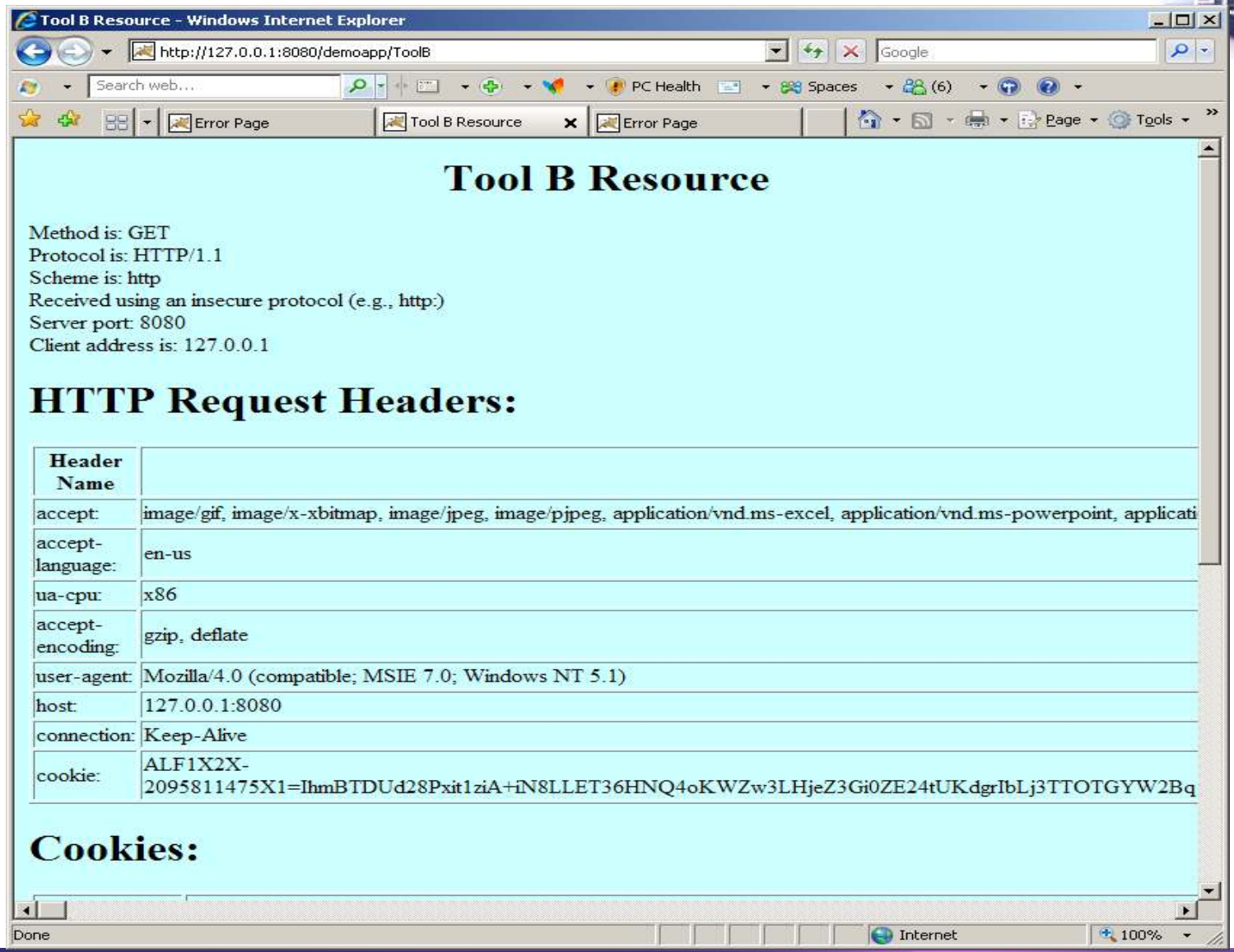
The user accesses the ToolA resource



12. User retains Token and can access Tool B because Gatekeeper accepts token



User accesses ToolB without challenge



Status

- Currently operational and used by Serena Business Mashup product
- Has been extended to work with C++ ISAPI applications
- Using native C# and .Net code to do the WS-Trust Request for Token, the general scheme (but not using the gatekeeper) has worked with .Net applications

Features and Extensions

- Remote data – for long query strings
- Master cookie – cross domain
- HTTPS-HTTP Redirect workaround

Big picture:
Some issues we hope Higgins can help supply technology to address

Crossing trust domains

- Passing the SAML token around a BPEL process works (sometimes with some BPEL engine work)
- But what happens when one of the tools involved in the BPEL orchestration is in a different trust domain and uses a different identity scheme?
  - E.g. Mainframe requiring RACF/SAF userid and password
  - UserID is different from the one in the SAML token
- Possible approaches:
  - WS-Federation
  - Some form of “headless” Identity Selector (rule/profile driven to replace human)
    - But the BPEL engine would need to run that
    - And any called service that calls another service
- Somewhere, there needs to be a mapping of IDs to make this work
  - Do one or more of the Federation Servers?
  - Can the Higgins IdAS help join disparate ID stores?

Some RPs still need the password

- Despite the availability of a SAML token, many services still expect a username and password
  - Other than having a Federation Server maintain
    - User ID mappings
    - The password for the domain of the RP service
  - Is there a better approach?
    - How to securely store the password so it can be read

Questions?

Thank you!

- Brian Carroll, ALF Project Lead

- Email: bcarroll@serena.com
- Tel: +1-503-617-2436 (Pacific Time)