Advanced Client
Conor P. Cahill
Systems Technology Lab
Intel Corporation
 Disclaimer

This presentation discusses work-in-progress within the Liberty Alliance Technology Expert Group. The end result of the specification process MAY be different than what is discussed here.
A client is a piece of software that invokes or exposes a service.
Where can we find clients?
Features of a client

- Close to, or in the hands of, the end-user
- More restricted communications path
  - Addressability
  - Bandwidth
- The root of authenticated sessions
Evolution of Liberty related Clients

- Phase 1: Liberty Enabled Client/Proxy (LECP)
- Phase 2: Active Client
- Phase 3: Advanced Client (aka Intelligent Client)
- Phase 4: Robust Client
Evolution: LECP

- Liberty Enabled Client/Proxy
- Facilitate SSO and Federation operations
  - Especially IDP Discovery
  - Authentication Request Direction
- Browser plug-in and/or Proxy server
- Incorporated into SAML 2.0 as ECP
Evolution: Active Client

- AKA: LUAD
- Local Web Services Consumer (WSC)
  - Radio Service client
  - Calendar Service client
- Liberty ID-WSF Authentication Service
  - SOAP profile of SASL
    - Supports *any* authentication protocol
  - Enabled SSO into Web Services
The client as an extension of the IdP
  - Off-line and privacy enabling modes
  - Strong local authentication
Locally hosted/managed services
Reporting
Future: Robust client

- Provisioning (pulled into Advanced Client)
  - Over-the-wire/air distribution of client modules
  - Support for trusted environments
- Mobility
  - Moving service instances and/or client modules
- Strong Authentication
  - multi-factor
Advanced Client: SSO/Federation

- Trusted Module (TM)
  - Extension of an IdP
  - Usually in some form of protected environment
    - Closed
    - Tamper resistant
    - E.g.: SIM
  - Drive SSO and/or Federation operations
  - Able to manufacture and/or store assertions
  - Able to function when IdP is not present
TM: Design Considerations

- **Privacy**
  - not shared by many users
  - ID of TM could be a correlation handle
  - Same for any public key used by TM for security

- **Security**
  - Mostly out-of-scope for Liberty
  - Enable features necessary for secure distribution
TM: Single Sign On Assertions

- Local manufacture of Assertions by TM (Minting)
  - IdP Authorizes TM to manufacture Assns
  - IdP controls facets of Assn
  - Relying Party (RP) can verify delegation
  - Privacy Protected by using unique keys for each RP

- Long term storage of IdP Issued Assns (Hoarding)
  - IdP issues Assns to TM
  - TM chooses when one of those Assns used for SSO
TM Conceptual Environment

Device

TM Protected Area

TM Manager

TM

TM

TM

Browser+

App(s)

IdP

SP

Calendar
1. TM Requests Minting Assertions
2. IDP responds with Minting Assns
1. SP initiates SSO (AuthnRequest)
2. Browser “discovers” TMs
3. TM Manager returns TM EPR
4. Browser forwards AuthnReq to TM
5. TM Responds with AuthnReq for SP
6. Browser forwards response to SP – user is not SSO’d into SP
1. Cal Application “discovers” TMs
2. TM Manager returns TM EPR(s)
3. App requests token for Cal WSP
4. TM responds with token for WSP
5. App sends ID-WSF call with token to WSP
6. WSP responds with Cal info
Client Service Instance (CSI)

- Locally hosted service instance
  - E.g. Profile, Calendar, Payment, etc.
- May or may not be in a trusted environment
- Looks, feels, and acts like a typical ID-WSF or ID-SIS service
- Issues:
  - Privacy (location becomes correlation handle)
  - Availability/connectivity
SHPS: A remote partner

- **Service Hosting/Proxying Service**
- Hosts a remote instance of service
  - Full implementation of service
  - Synchronization with Client Service Instance (CSI)
  - CSI seen as master, but WSCs interact with Hosted service
- **Proxies remote service invocations**
  - Forwards each invocation to CSI
CSI normal ID-WSF invocation

1. WSC discover’s Calendar service
2. DS returns CSI’s EPR to WSC
3. WSC invokes Calendar CSI
4. Calendar CSI returns data
CSI with SHPS hosting

1. WSC discovers Calendar service
2. DS returns Calendar EPR to WSC
3. WSC invokes Calendar Svc
4. SHPS Cal Svc returns response
1. WSC discover’s **Calendar** service
2. DS returns Calendar EPR to WSC
3. WSC invokes Calendar Svc
4. SHPS forwards req to Cal CSI
5. Cal CSI sends response to SHPS
6. SHPS returns response to WSC
Current Status

- **Advanced client**
  - Requirements completed
  - Specifications in development
  - 1Q2007 public draft

- **Robust Client**
  - Requirements completed
  - Some specifications work (Provisioning)
  - No public estimates as to spec release
More Information

- Web: http://www.projectliberty.org
- My blog: http://conorcahill.blogspot.com
- Email: Conor.P.Cahill – at - intel.com