WS-Federation 1.1 Overview

OASIS WSFED TC inaugural meeting
June 6-7, 2007

Agenda

1. Introduction
2. WS-Trust extensions for federations
3. STS service model extensibility
4. Federation metadata
5. Federated sign-out and Web requestors
6. Summary

Agenda Part 1

Introduction

- Vision and Goals
- Basic Terminology and Components
- Sample Federation Scenarios

Vision: Extend WS-Trust

- Flexible identity federation architecture
  - Clean separation between trust mechanisms, security token formats, and token protocol
  - Infrastructure supports browser & SOAP requestors
- Simplified configuration
  - Federation metadata to fill gaps in policy
  - Federation partners can automate configuration
- Reusable token service model
  - Common claims interface for attributes, pseudonyms & authorization data

Promise: Finish the Roadmap

- Federation vision declared 5 years ago
- Web Services security stack roadmap
  - Set of composable specifications to enable broad range of secure Web Services solutions
  - All specifications to be ratified by industry through open standards process
- WS-Federation completes the promise to finish the roadmap

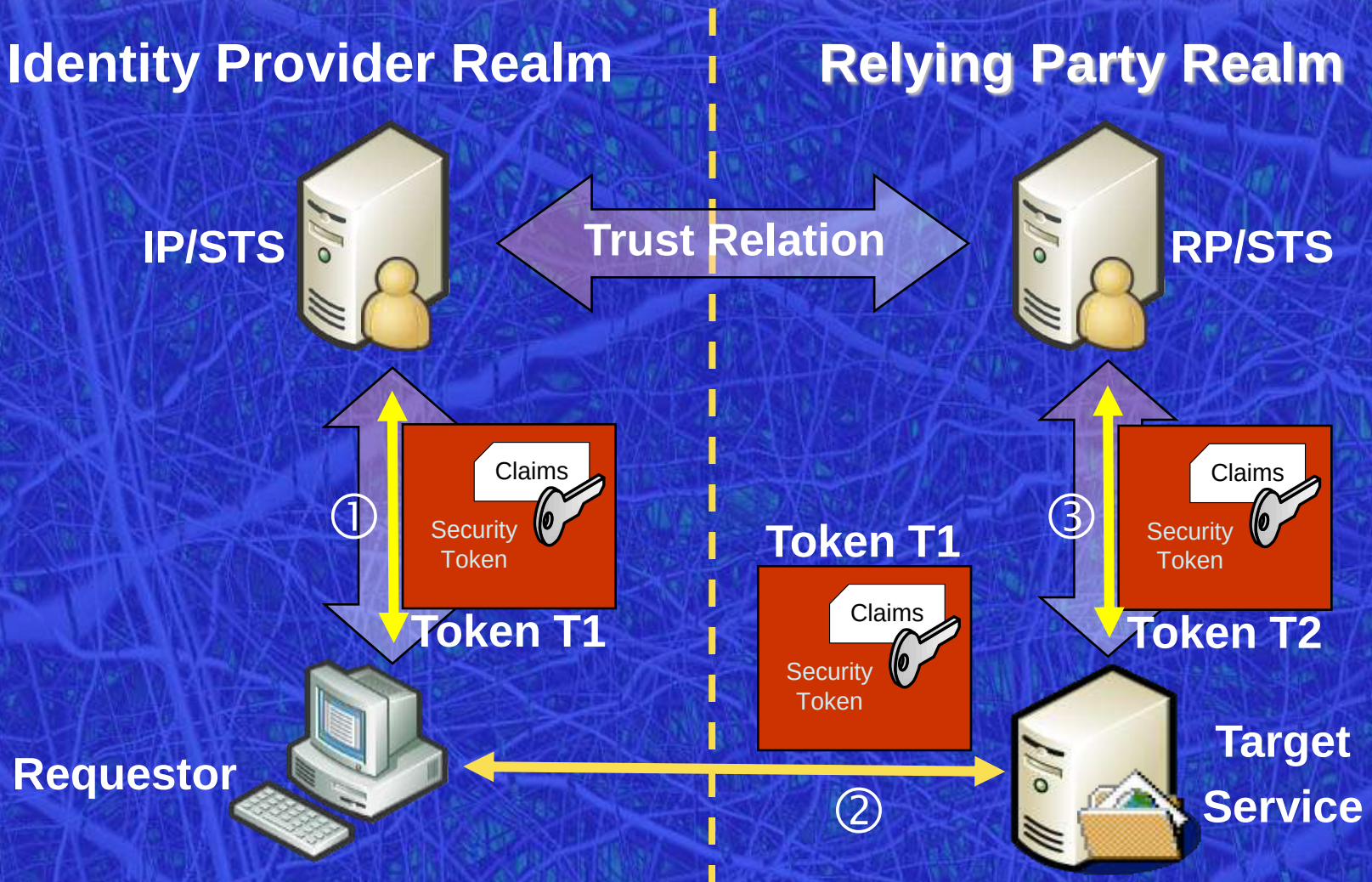
Goals and Requirements

- Promote identity federation
  - Enhance WS-Trust STS support for distributed authentication and authorization across realm boundaries
  - Make identity mapping optional (for privacy or personalization)
  - Enable different levels of privacy for different types of personally identifying information
- Reduce operational friction in federations
  - Support mix & match of trust topologies and token types
  - Enable automated configuration using Federation Metadata
  - Allow single infrastructure to serve both SOAP and Web requesters
- Reuse the WS-Trust STS model
  - Offer common interface for broad range of federation services
  - Allow identity, authentication, and authorization data to be shared as claims without requiring a specific token type

Basic Terminology

- ***Requestor*** – A programmatic agent for obtaining information or service
- ***Subject*** – The entity on whose behalf a Requestor operates
- ***Claims*** – Statements made about a subject
- ***Security Token*** – A data structure for expressing collections of claims
- ***Security Token Service (STS)*** - A Web service that provides issuance and management of security tokens
- ***Identity Provider (IP)*** – An entity, typically a trusted third party authority, that provides claims about a set of Subjects
- ***IP/STS*** – STS operated by an IP to issue claims using tokens
- ***Relying Party (RP)*** – An entity that provides information or services to Requestors based on claims they present
- ***Target Service*** – A web service (or application) operated by an RP
- ***RP/STS*** – STS operated by a RP to issue claims using tokens

Basic Components

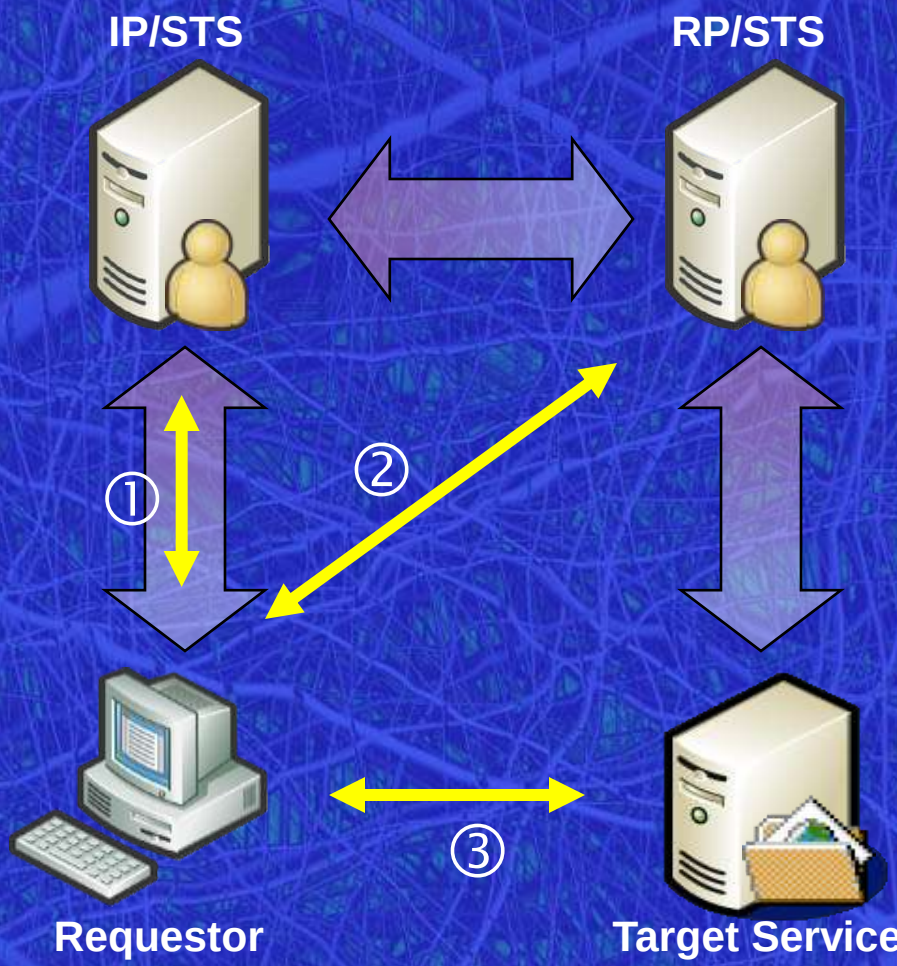


Federation Scenarios

- The following are sample federation scenarios depicting trust topologies and claims flow
- They are not comprehensive or prioritized
- There are other valid scenarios

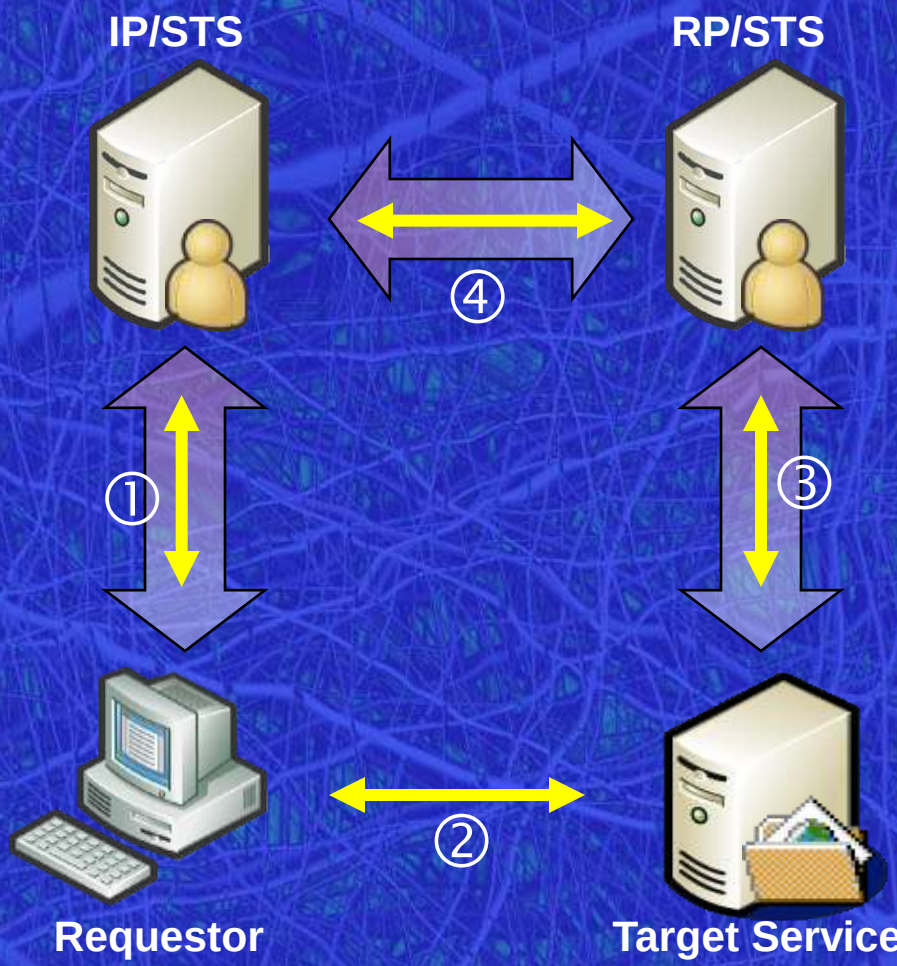
Federation Scenarios

Direct Trust & Token Push



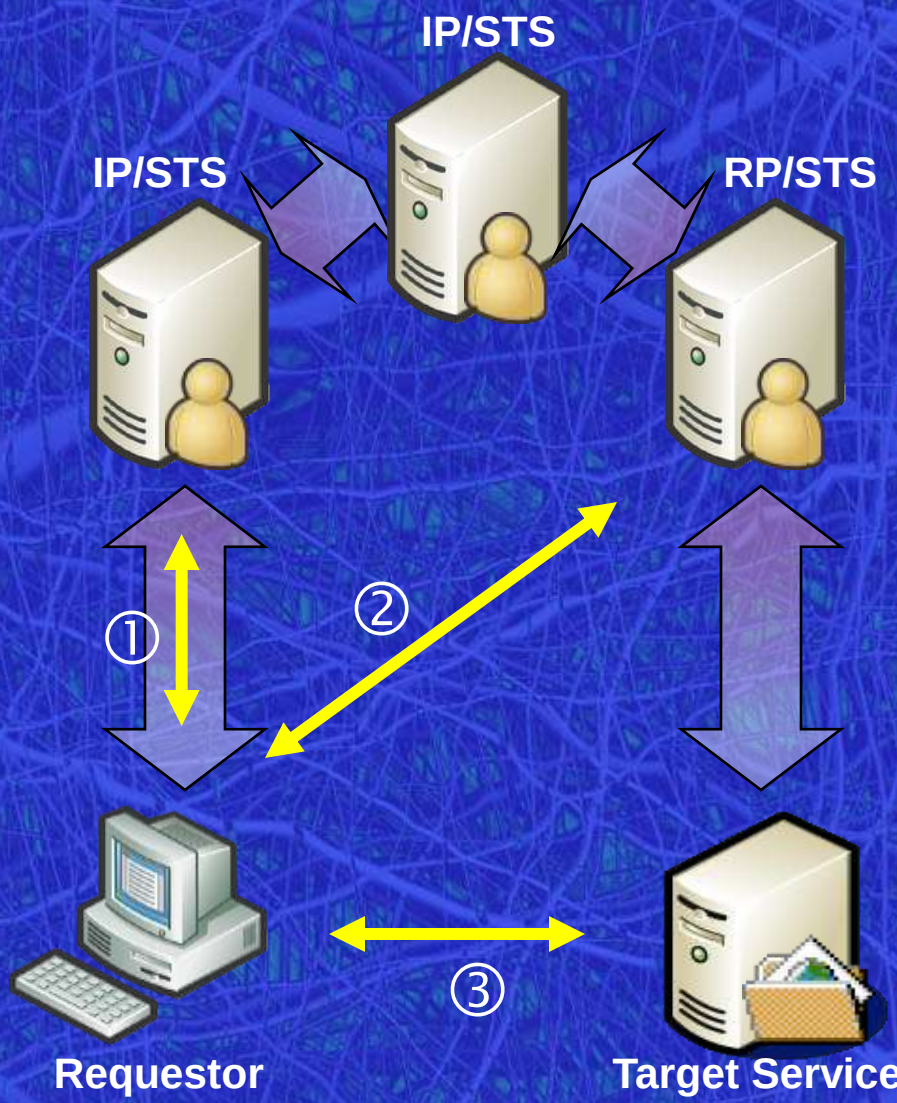
Federation Scenarios

Direct Trust & Token Pull



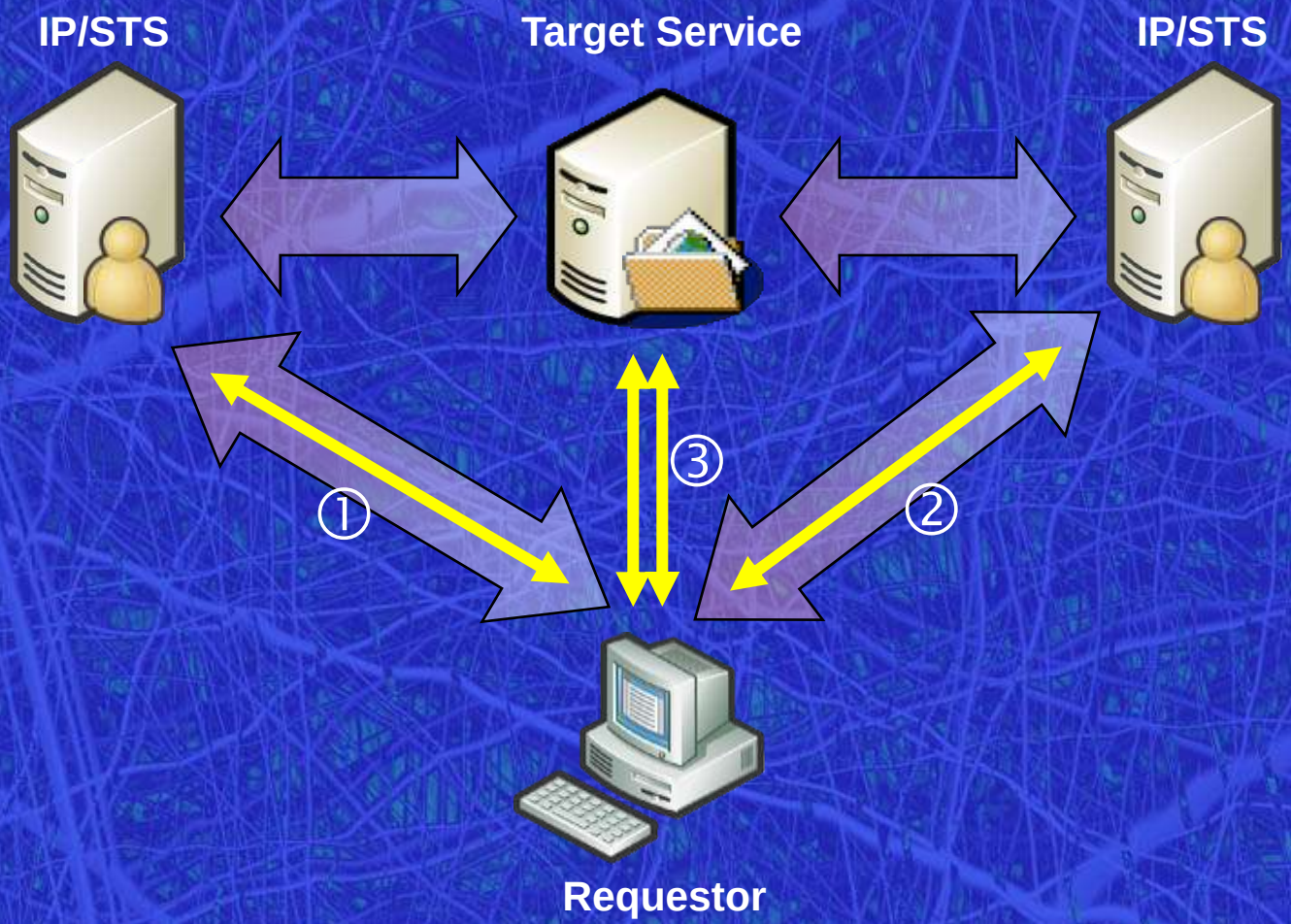
Federation Scenarios

Indirect Trust



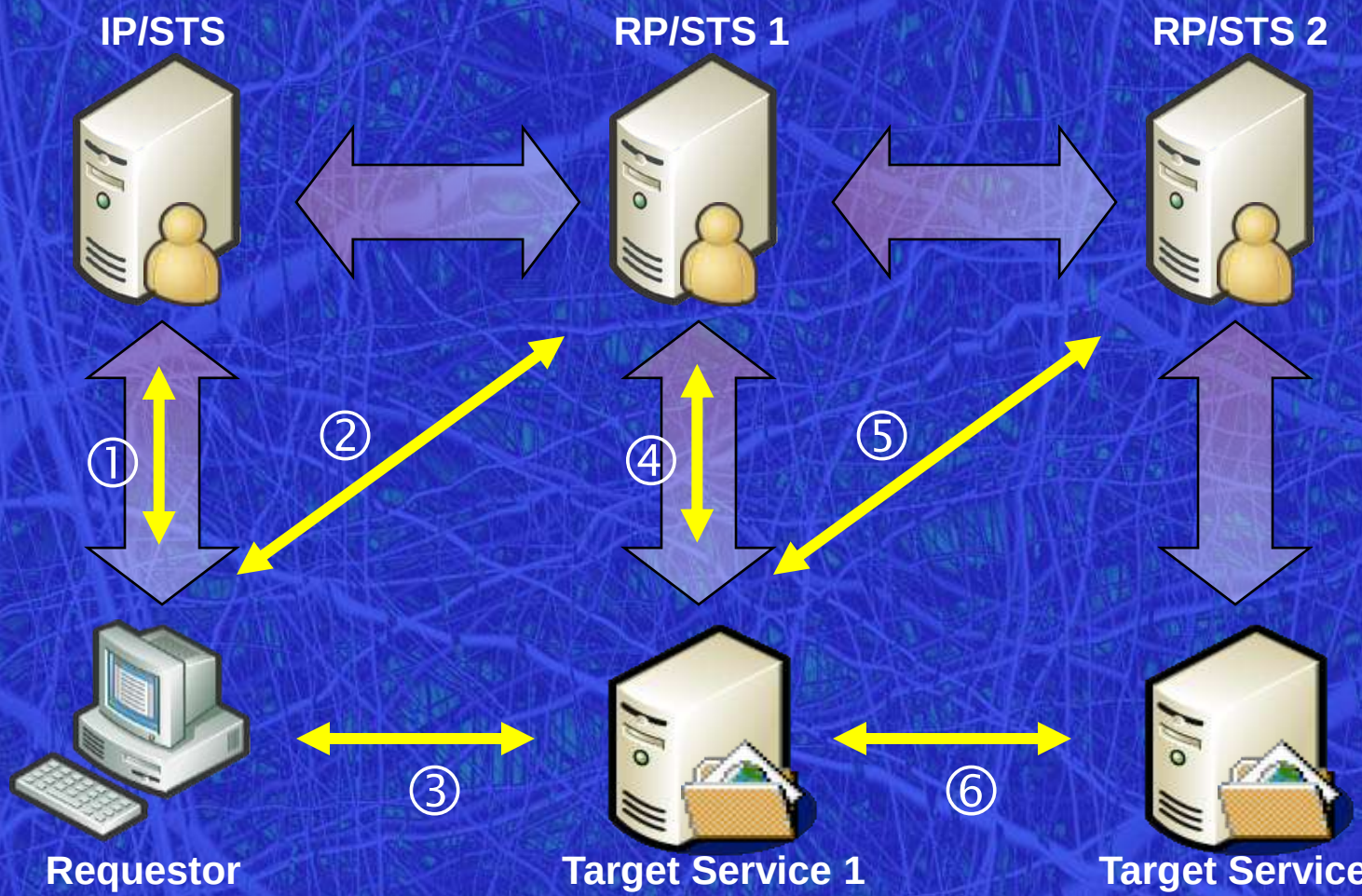
Federation Scenarios

Multiple Tokens with Direct Trust



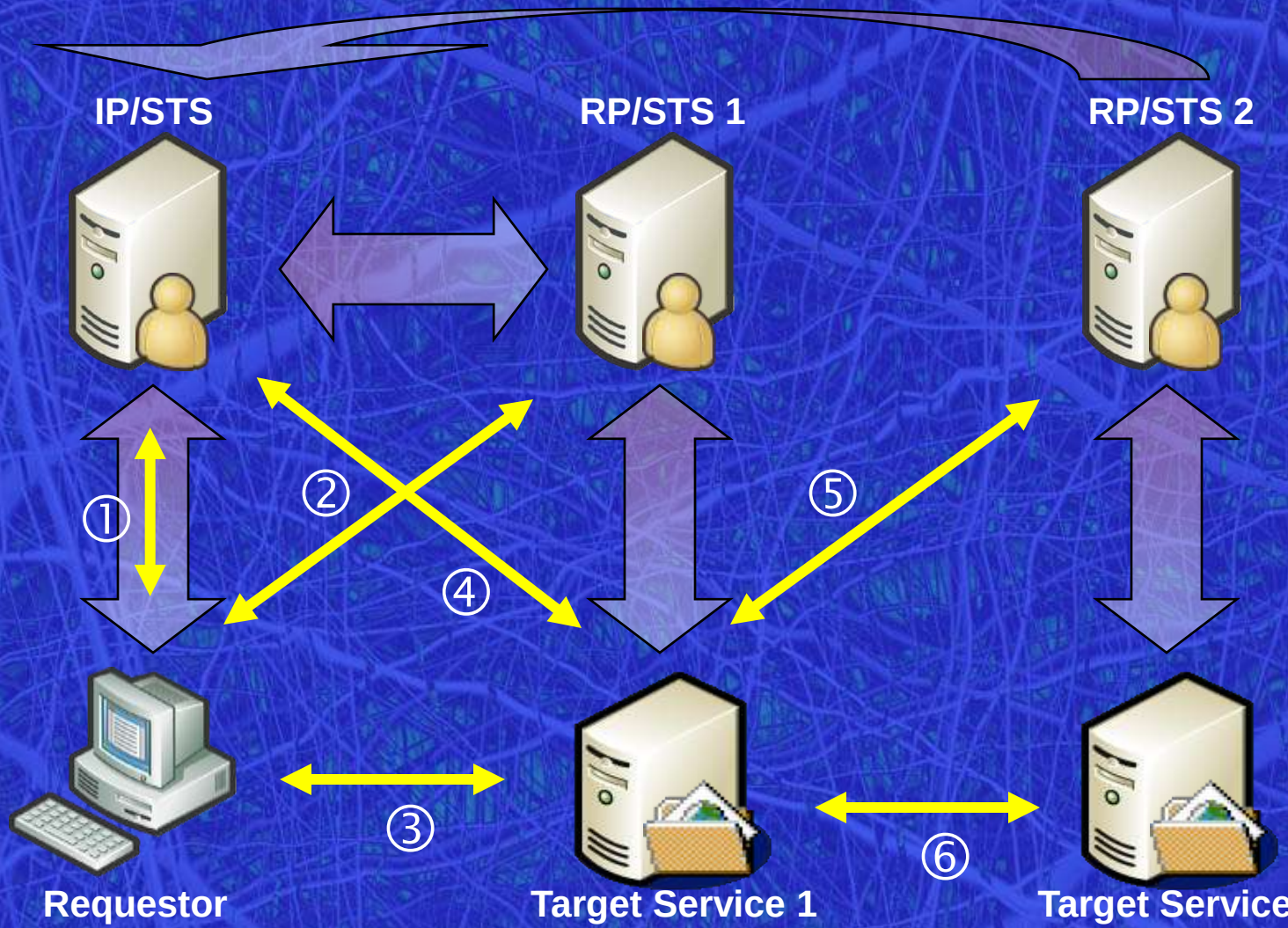
Federation Scenarios

Delegation with Indirect Trust



Federation Scenarios

Delegation with Direct Trust



Agenda Part 2

WS-Trust Extensions for Federations

- Token and Protocol Extensions
  - Reference tokens
  - Identifying Federations
  - Validation & Proof Tokens
  - Client-Based Pseudonyms
  - Token Freshness
- Privacy

WS-Trust Extensions

Indicating Federations

- STSs that participate in multiple federations need a way to distinguish the federation for which a request applies
  - Could use different endpoints
  - Can provide a parameter to the RST using new extension
    - <fed:FederationID ...>xs:anyURI</fed:FederationID>

WS-Trust Extensions

Reference Tokens

- Indicates where to obtain actual tokens
  - <fed:ReferenceToken ...>
- Can be used with WS-Security
- Assertion for use with WS-SecurityPolicy
- Allows multiple locations for the token
- Allows verification information about the token

WS-Trust Extensions

Proof Tokens from Validation

- Often trust between federated partners is actually between the corresponding STSs
- Target Services don't know the key-transfer-key
- Extension formalizes how Target Services get the session key from their STS

WS-Trust Extensions

Freshness Requirement

- RP may have policy indicating that an STS should only accept credentials of a specific age when issuing tokens for the RP
  - <fed:Freshness AllowCache="xs:boolean" ...>
- Extension can specify this limit in the RST, and if cached credentials can be used

WS-Trust Extensions

Authentication Types

- RP may have policy that an STS should only accept credentials of specific authentication types when issuing tokens for the RP
- WS-Trust provides a mechanism, but no defined values
- Extension defines several commonly used values

Privacy

- WS-Federation addresses three specific areas of concern for privacy in federated scenarios:
  1) Confidential tokens
  2) Parameter confirmation
  3) Obtaining privacy statements

Privacy

Confidential Tokens

- WS-Trust does not define specific rules for mandating claim confidentiality
- WS-Federation defines a parameter to RST that indicates which claims are requested to be protected
  - <priv:ProtectData ...>
- Any claim dialect can be used

Privacy

Parameter Confirmation

- WS-Trust does not request that RST parameters be honored or that selected values be returned in RSTR
- These extensions (when used) require the STS to:
  - Include in the RSTR the values used for specified parameters
  - Fault if a parameter in the RST is not used
  - Return claims put in the issued token

Privacy

Obtaining Privacy Statements

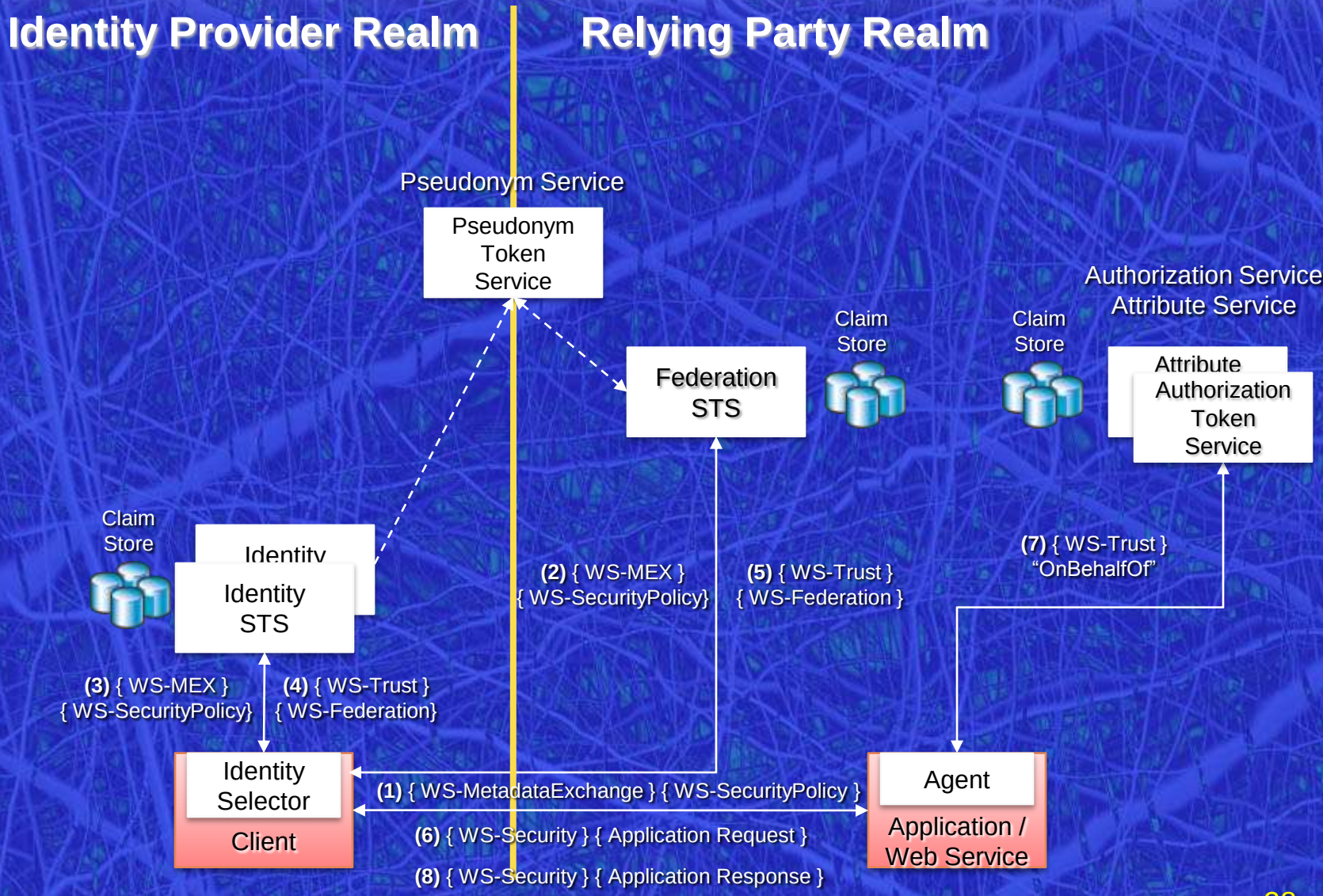
- The specification does not define the contents; only the mechanism
  - How to use WS-Transfer
  - How to use WS-MetadataExchange
  - How to use HTTP

Agenda Part 3

STS Service Model Extensions

- Extended Token Service Model
- Attribute Service
- Authorization Service
- Pseudonym Service

Extended Token Service Model



Attribute Service

STS as Attribute Service

- An IP/STS or RP/STS can function as an Attribute Service
  - Attributes are claims
  - Tokens carry claims
  - STS can provide normalized I/F to any repository
- Attributes obtained based on policy or explicit request
  - Inline claim transformation
  - Explicit claim transformation

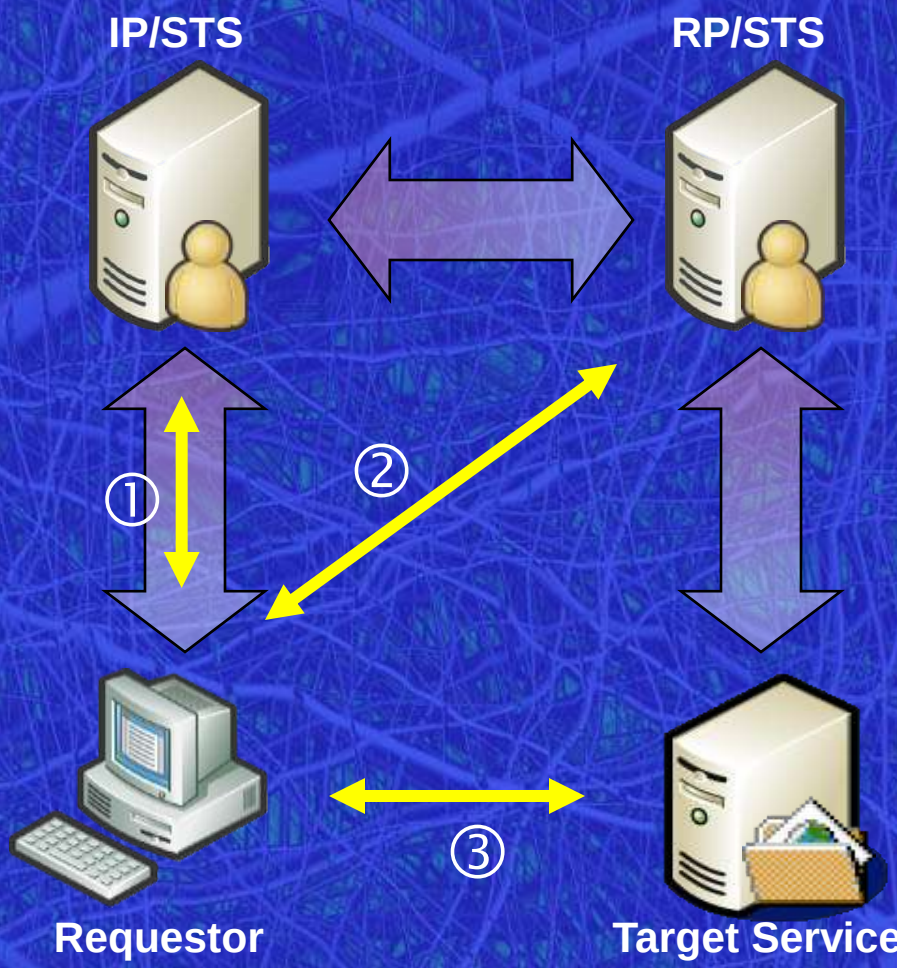
Attribute Service

Inline Claim Transformation

- WS-Security Policy
  - Target Service policy:
    - RP/STS as issuer with Application claim types
  - RP/STS policy:
    - IP/STS as issuer with Federation claim types
- Requestor automatically delivers correct claims
  - IP/STS issues token with Federated claim types
  - RP/STS issues token with Application claim types

Attribute Service

Inline Claim Transformation



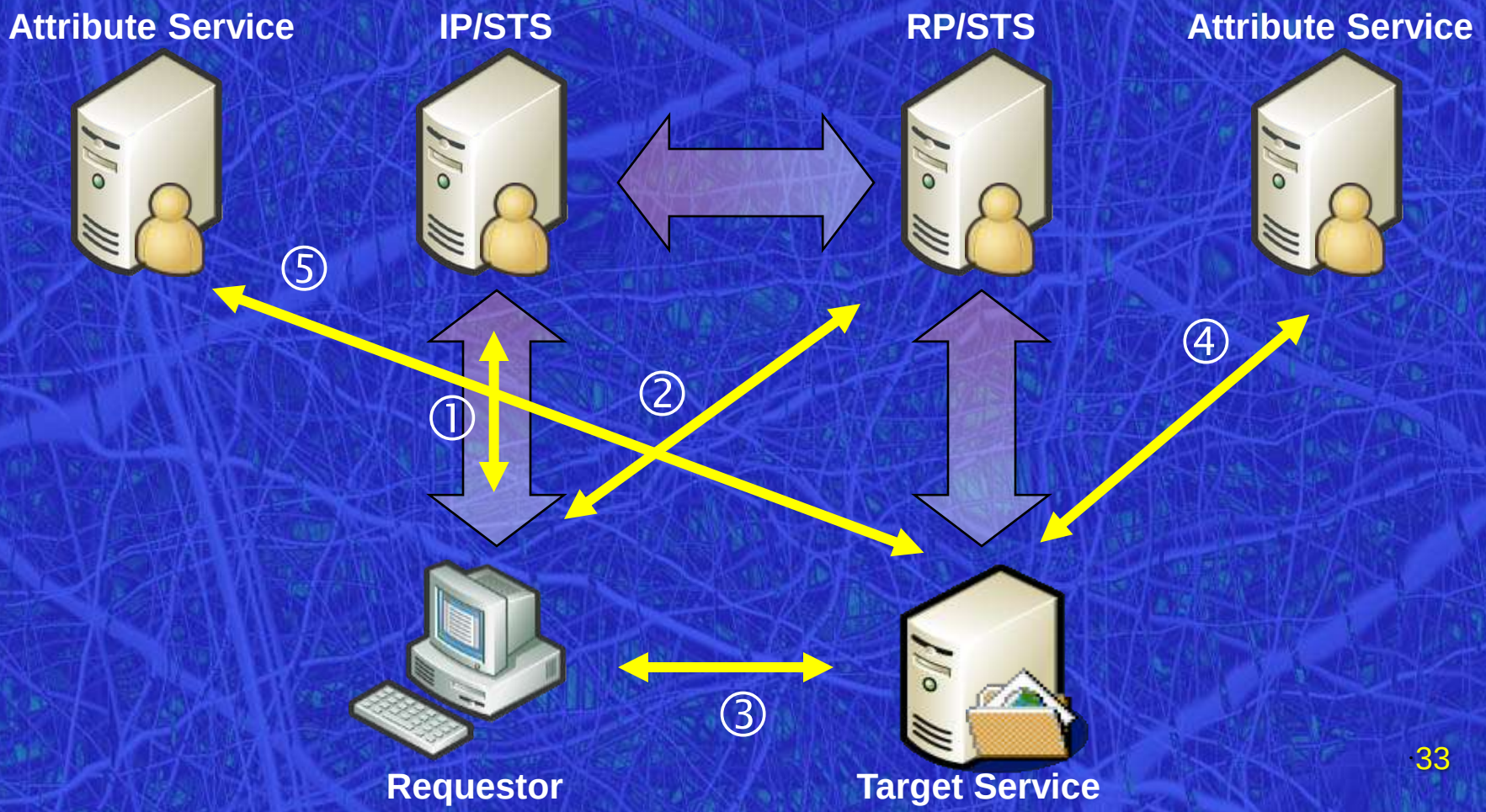
Attribute Service

Explicit Claim Transformation

- Attribute service interfaces
  - RST Issue to Target Service policy:
  - RST Issue to Target Service OnBehalfOf user
- Simplifies application programming
  - Target Service gets claims without
    - Writing LDAP, SQL or special repository code
    - Mapping from repository schema and namespace
    - Maintaining credentials for repositories
    - Being authorized for direct access to repositories

Attribute Service

Explicit Claim Transformation



Pseudonym Service

- Pseudonyms are different “personas” of an identity
- Pseudonym Service
  - Performs mapping between personas
  - Logically just a special type of Attribute Service
  - Can be invoked by client, IP, or RP
- Supports different usages of personas
  - Global, Pair wise, Random, …

Pseudonym Service

Pseudonym Management Operations

Operations

- Create pseudonyms for target
- Get pseudonyms matching filter
- Update pseudonyms for target
- Delete pseudonyms for target

- Filters
  - Specify subset of pseudonyms for operations
  - Pass filters in WS-ResourceTransfer
  - Pass filters in EPR reference properties

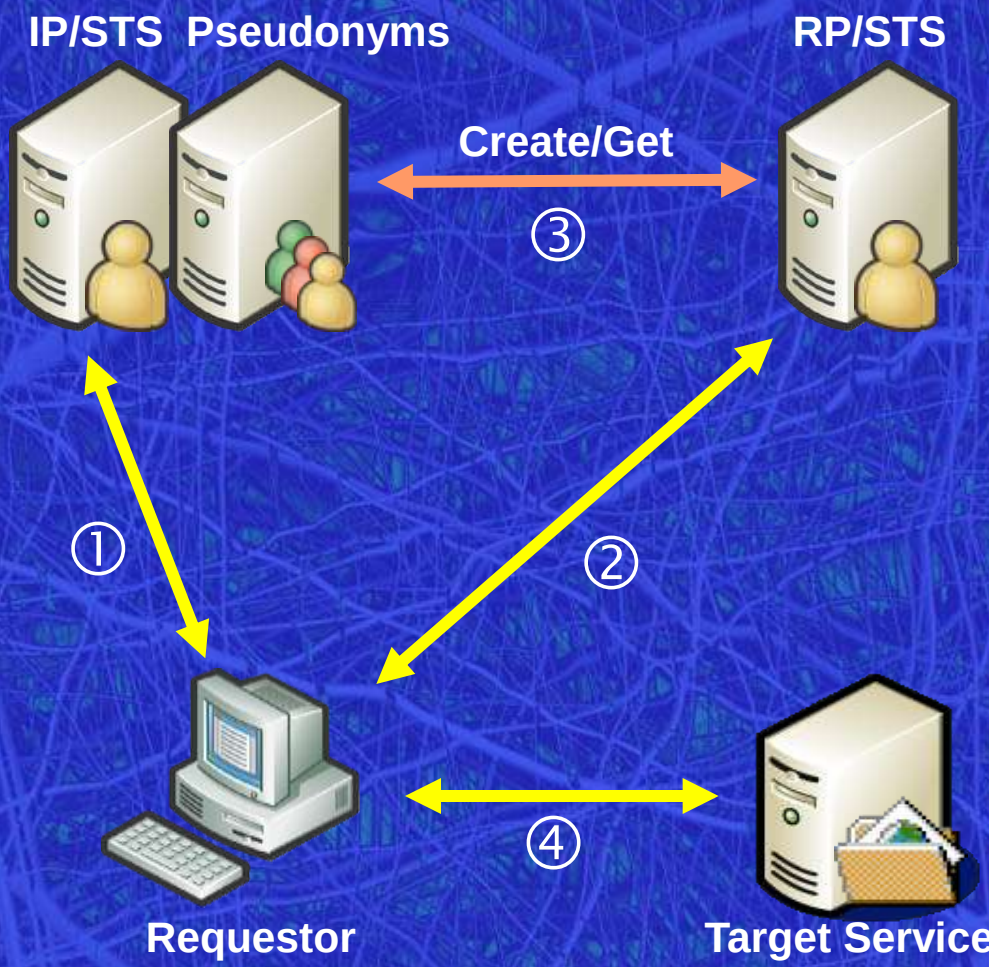
Pseudonym Service

RST Extensions for Pseudonym Retrieval

- Pseudonym service often part of STS
- Request pseudonyms
  - fed:RequestPseudonym/@Lookup
  - fed:RequestPseudonym/@SingleUse

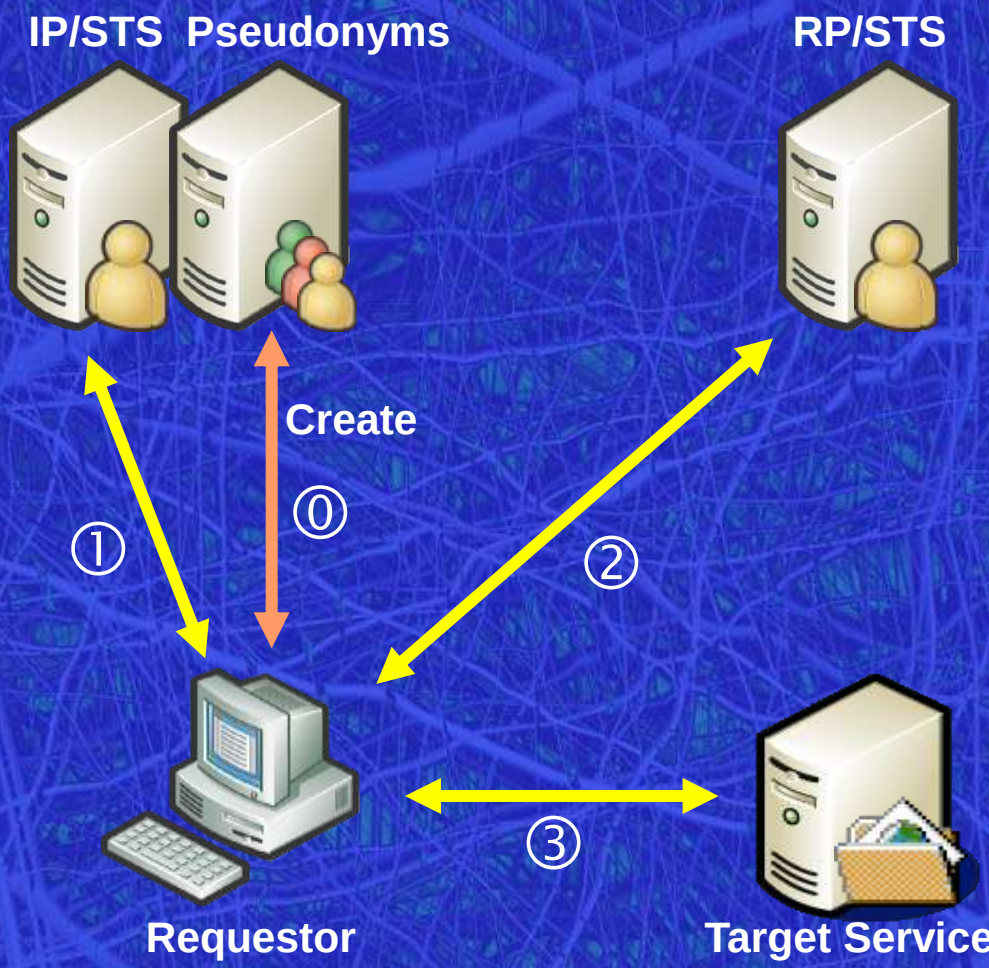
Pseudonym Service

RP-managed Pseudonym (Identity Mapping)



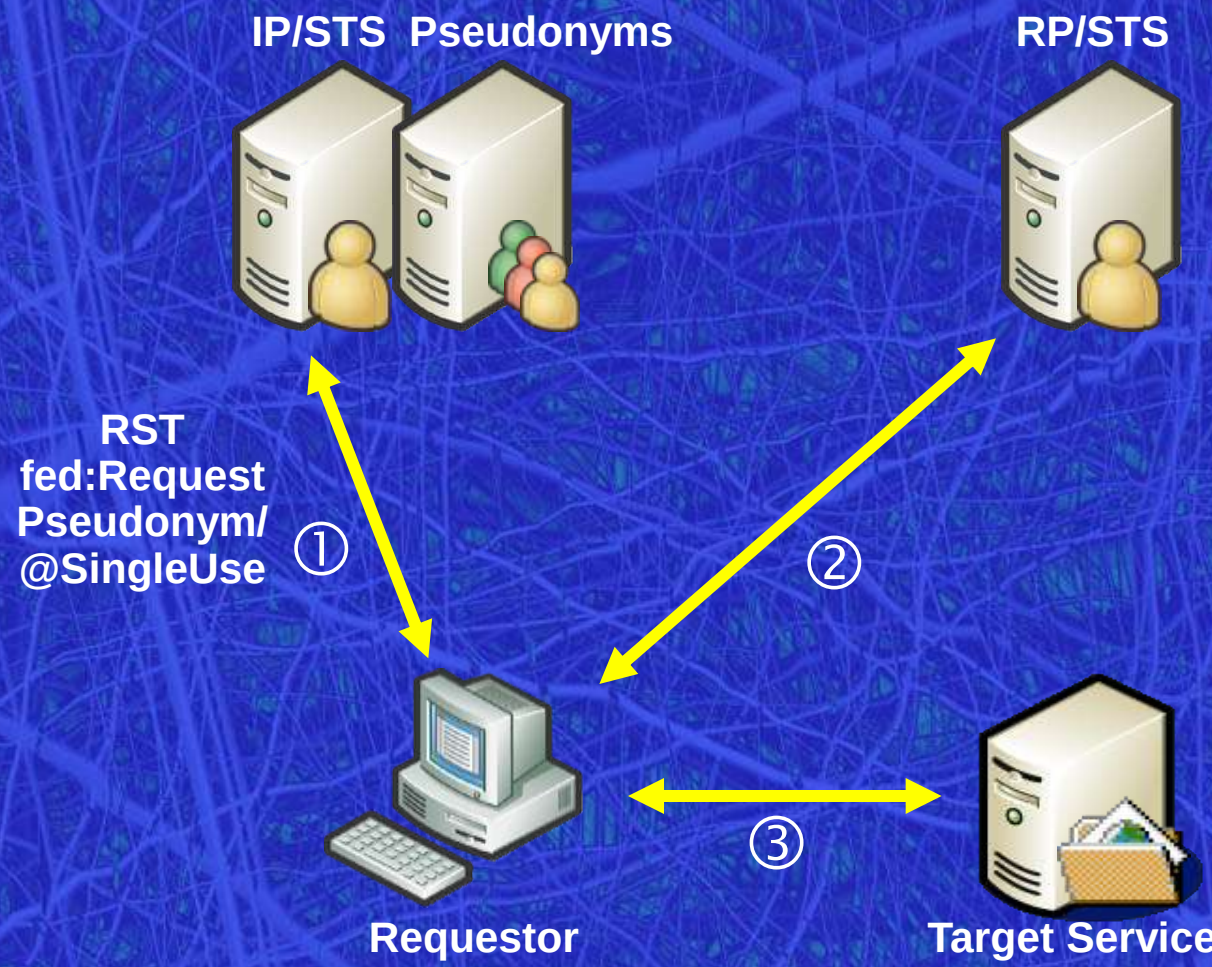
Pseudonym Service

Pre-registered Pseudonym for RP



Pseudonym Service

Random, Single-use Pseudonyms



Pseudonym Service

RST Extensions for Client-Based Pseudonyms

- Clients can specify PII to use as basis for pseudonyms
- Clients can specify PII to include in token
  - ID
  - Display Name
  - Email
  - …

Authorization Service

- An Authorization Service may be implemented as a dedicated STS
  - Configured with detailed knowledge of the access policy requirements of Target Services
- WS-Federation defines the following to facilitate federated authorization
  - A common processing model & requirements
  - An authorization context
  - A common claim dialect
  - Associated policy assertions

Authorization Service

Processing Model

- Logical Requirements Table:
  - EPR for the target service
  - Reference properties from the target service EPR
  - Parameters of the RST
  - External access control policies
- Logical Claim Table:
  - Proven claims bound to RST
  - Supplemental context information
  - External authorization policies

Authorization Service

STS Processing Requirements

- Must accept AppliesTo
- Must specify AppliesTo in RSTR
- Should encode AppliesTo in issued tokens
  - AppliesTo in token may be broader than requested
- Must accept reference properties
- Must accept common claim dialect
- Must accept additional context
- May ignore context items it doesn't recognize

Authorization

Authorization Context

- A set of <ContextItem> elements, each has:
  - URI name of the item
  - Optional URI scope of the item
    - E.g. Requestor, Target, Action, …
  - Optional string value

Authorization

Common Claim Dialect

- A syntax for constructing/parsing claims
  - Does not specify claim semantics or namespace
- A set of <ClaimType> elements, each has:
  - URI indicating type of claim
  - Mandatory/optional flag
  - Optional string value

Authorization

Policy Assertions

- RequiresGenericClaimDialect
- AdditionalContextProcessed

Agenda Part 4

Federation Metadata

- Metadata documents
- Metadata statements
- Obtaining metadata documents

Service-specific Metadata

- Dynamic request retry

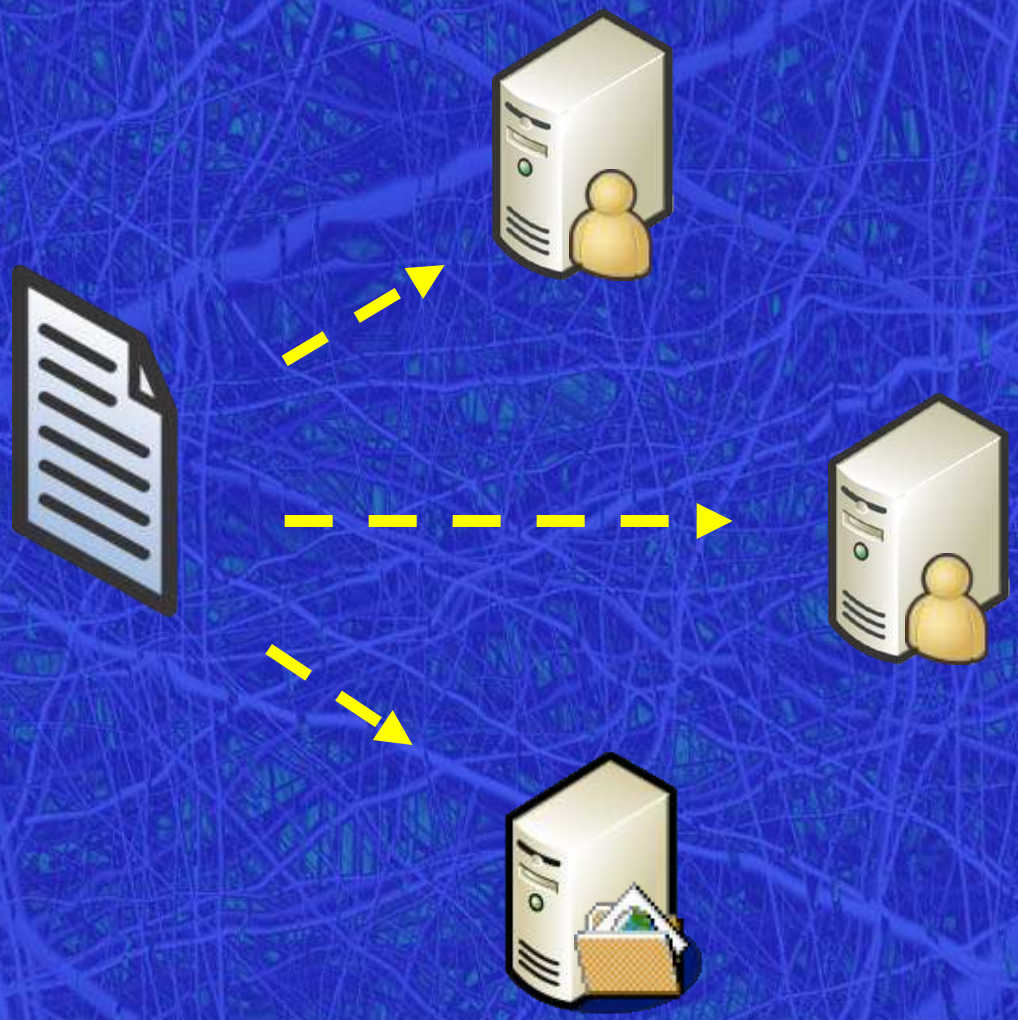
Federation Metadata

Metadata Documents

```
<fed:FederationMetadata xmlns:fed="..." ...>
    <fed:Federation [FederationID="..."] ...>
        <mex:MetadataReference>
        </mex:MetadataReference>
    </fed:Federation>
    <fed:Federation [FederationID="..."] ...>
        [Federation Metadata Statements]
    </fed:Federation>
    [Signature]
</fed:FederationMetadata>
```

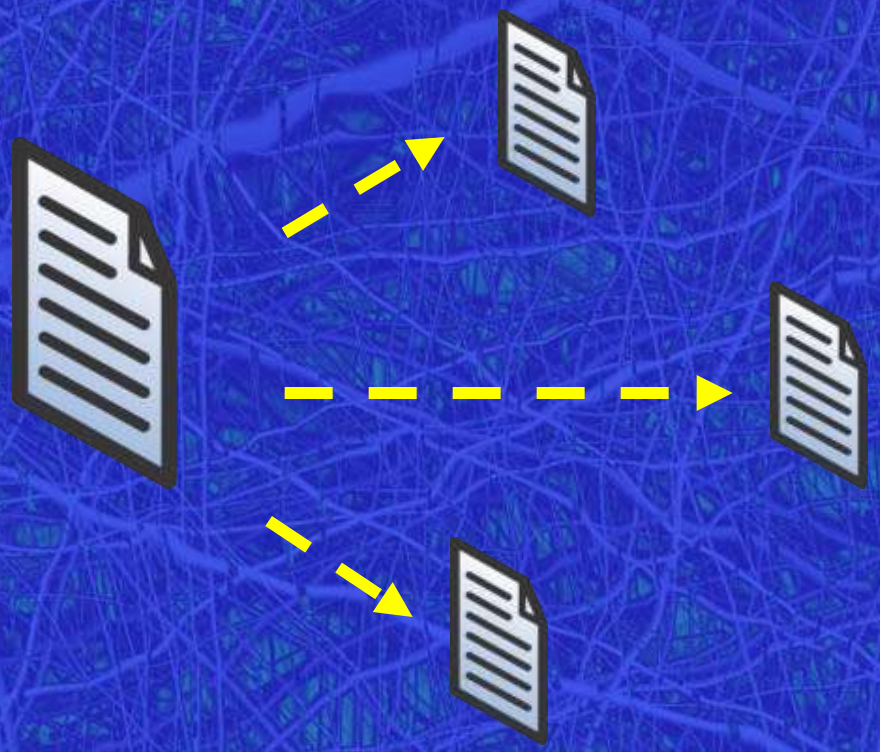
Metadata

Simple Metadata Document



Metadata

Compound Metadata Document



Federation Metadata

Metadata Statements

- TokenSigningKeyInfo
  - The key/token used to sign issued tokens
- TokenKeyTransferKeyInfo
  - The key/token to use when transferring keys/secrets
- IssuersNamesOffered
  - List of logical names with which a STS is associated
- TokenIssuerName
  - Logical name of the associated STS
- TokenIssuerEndpoint
  - Endpoint of the associated STS
- PseudonymServiceEndpoint
  - Endpoint of the associated pseudonym service

Federation Metadata

Metadata Statements

- AttributeServiceEndpoint
  - Endpoint of the associated attribute service
- SingleSignOutSubscriptionEndpoint
  - Endpoint to which sign-out notification subscription requests are sent
- SingleSignOutNotificationEndpoint
  - Endpoint to which manual Sign-out messages should be sent
- TokenTypesOffered
  - List of token types a STS can issue
- UriNamedClaimTypesOffered
  - List of claims types a STS can issue, display name and description
- AutomaticPseudonyms
  - STS automatically applies pseudonyms

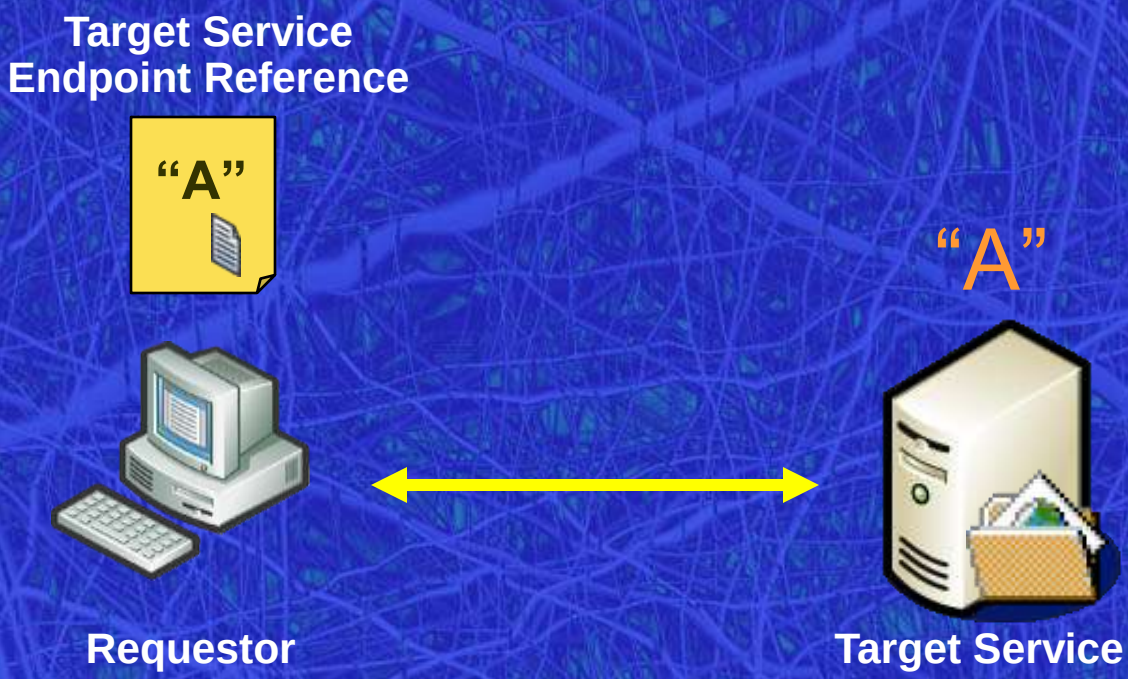
Federation Metadata

Obtaining Metadata Documents

- Several ways to obtain metadata documents
  - HTTP/S GET from well-known URLs
  - DNS SRV records
  - WS-Transfer/WS-ResourceTransfer
  - WSDL embedding
  - WS-MetadataExchange
- Secure request methods are preferred

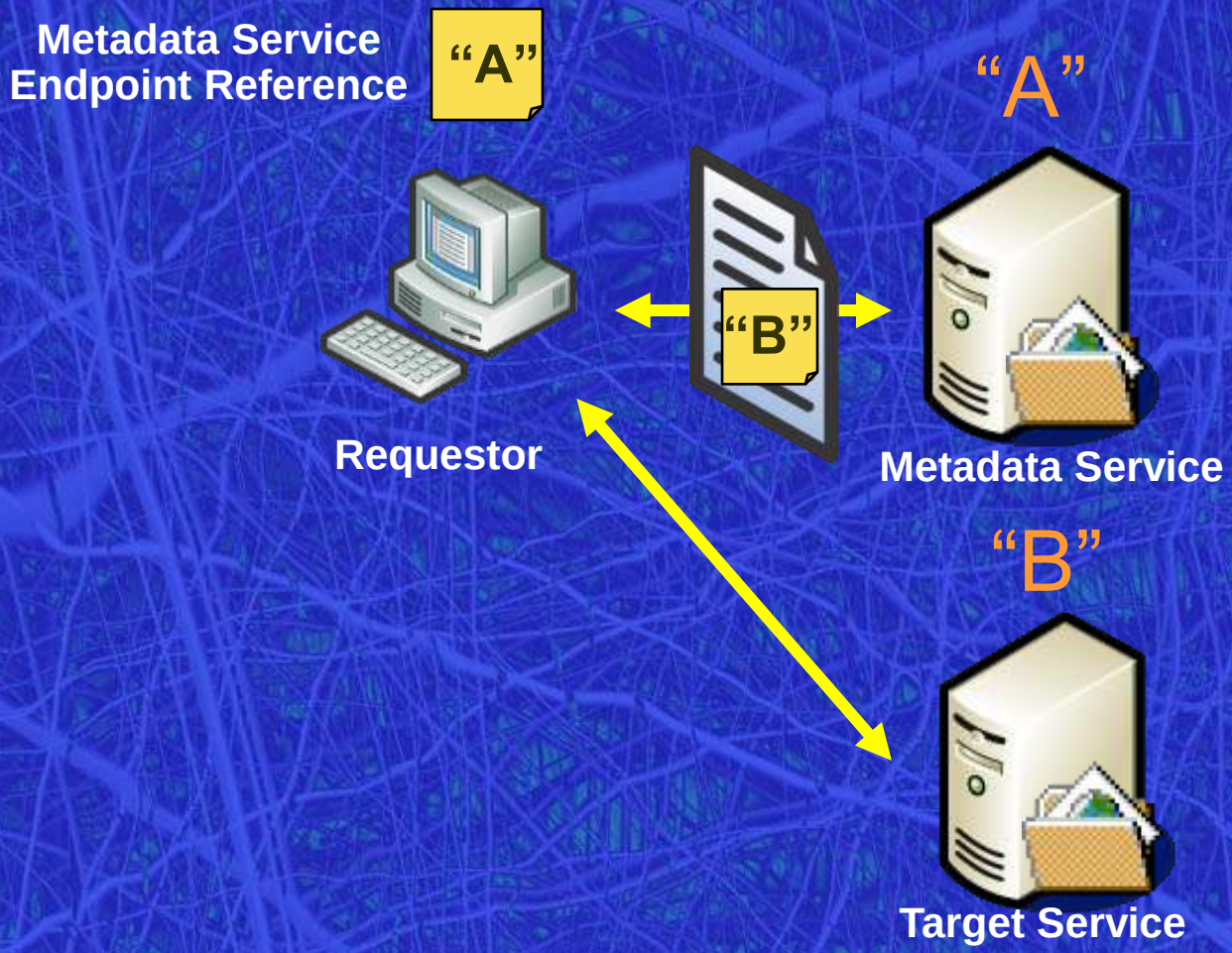
Federation Metadata

Metadata Embedded in Target Service EPR



Federation Metadata

Metadata Service Publishes Target Service Metadata



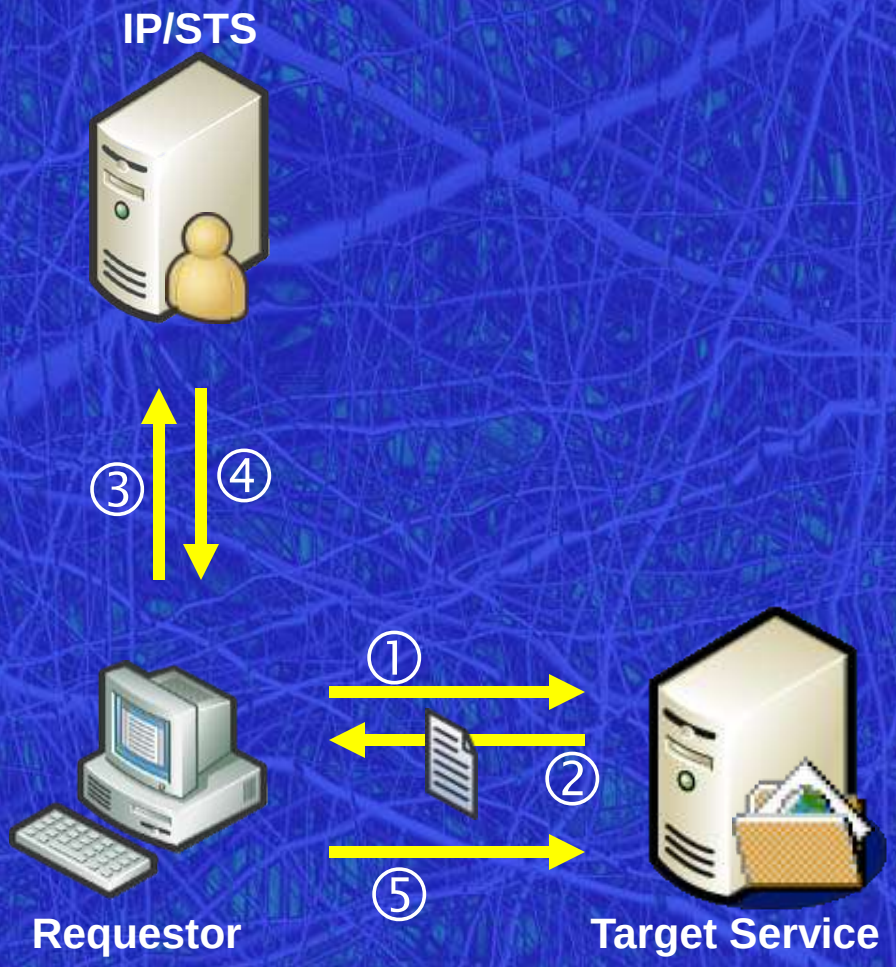
Service-specific Metadata

Dynamic Request Retry

- Not all policy/metadata can be expressed statically
- WS-Federation introduces a SOAP Fault to indicate policy/metadata specific to a request
- This Fault formalizes returning WS-MetadataExchange structures
- IssuesSpecificMetadataFault assertion allows indication of support in policy

Service-specific Metadata

MetadataExchange structures in SOAP Fault



Agenda Part 5

Federated Sign-Out

- Sign-Out concepts
- Federated sign-out

Web Requestors

- General model
- HTTP binding
- Message flows
- Request & result references
- Home realm discovery
- Interoperability baseline

Sign-out

Concepts

- *Sign-in* establishes an identity used to obtain credentials for a set of target sites
- *Sign-out* terminates the use of the identity and the associated target site credentials (and optionally cached state)
- The *sign-out* process is optional since credentials have limited life-times
- *Sign-out* is different from canceling since it applies to all tokens obtained for the target sites

Federated Sign-out

Mechanisms

- *Initial Sign-out* message
  - Sent by Requestor
  - Sent to IP STS or RP
- *Federated Sign-out* messages
  - RP forwards to IP STS if necessary
    a) IP STS sends explicit msgs to all RPs where the credentials apply
    b) IP STS publishes sign-out notification

Web Requestors

- WS-Federation defines a serialization for use with Web Browsers
  - Functionally equivalent to SOAP bindings
  - Optimizations for Web browser usage
- Supports push and pull models
- Supports GET and POST
- Basic home realm discovery
- Defines a base functionality set

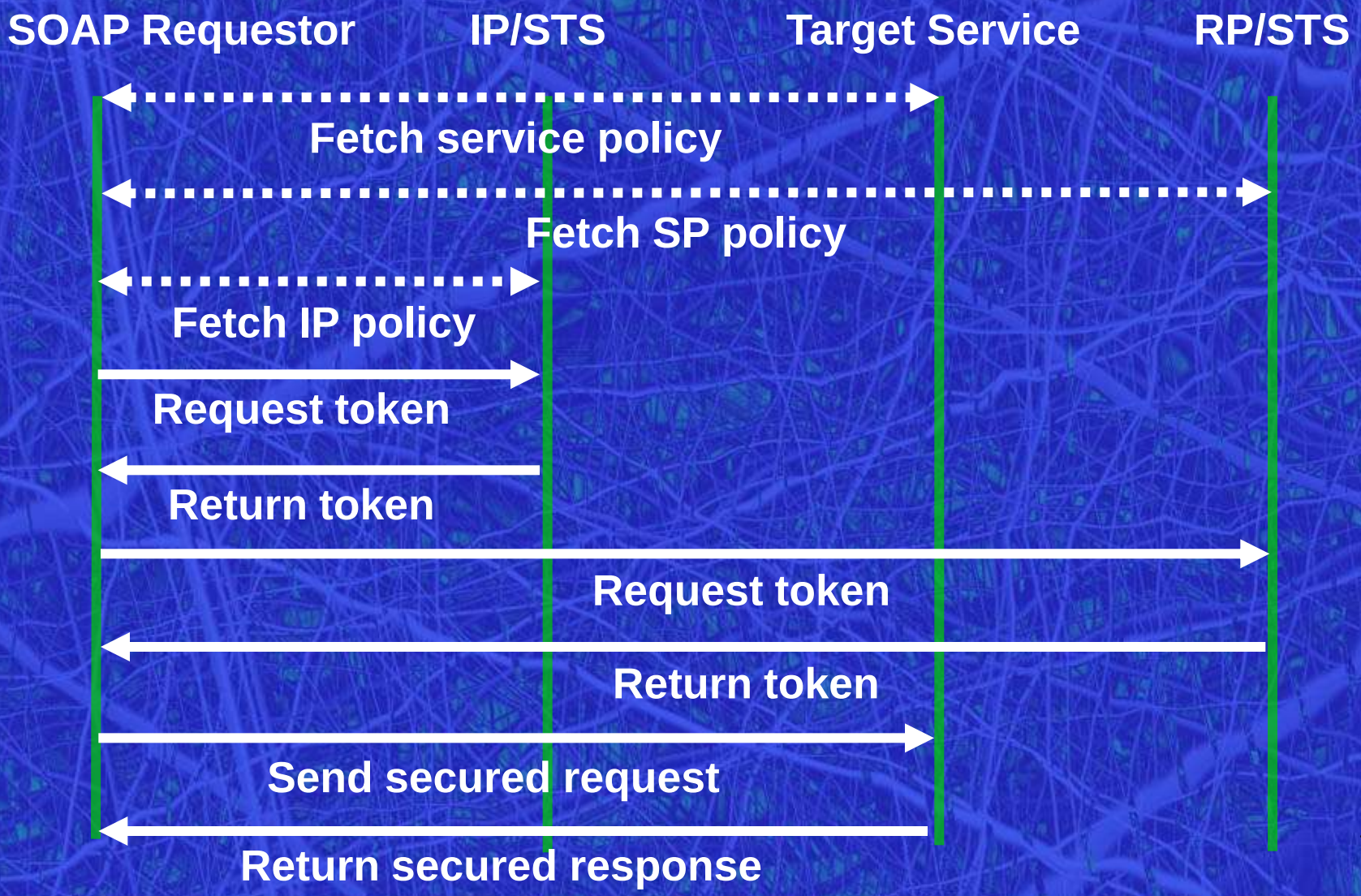
Web Requestors

Drilldown

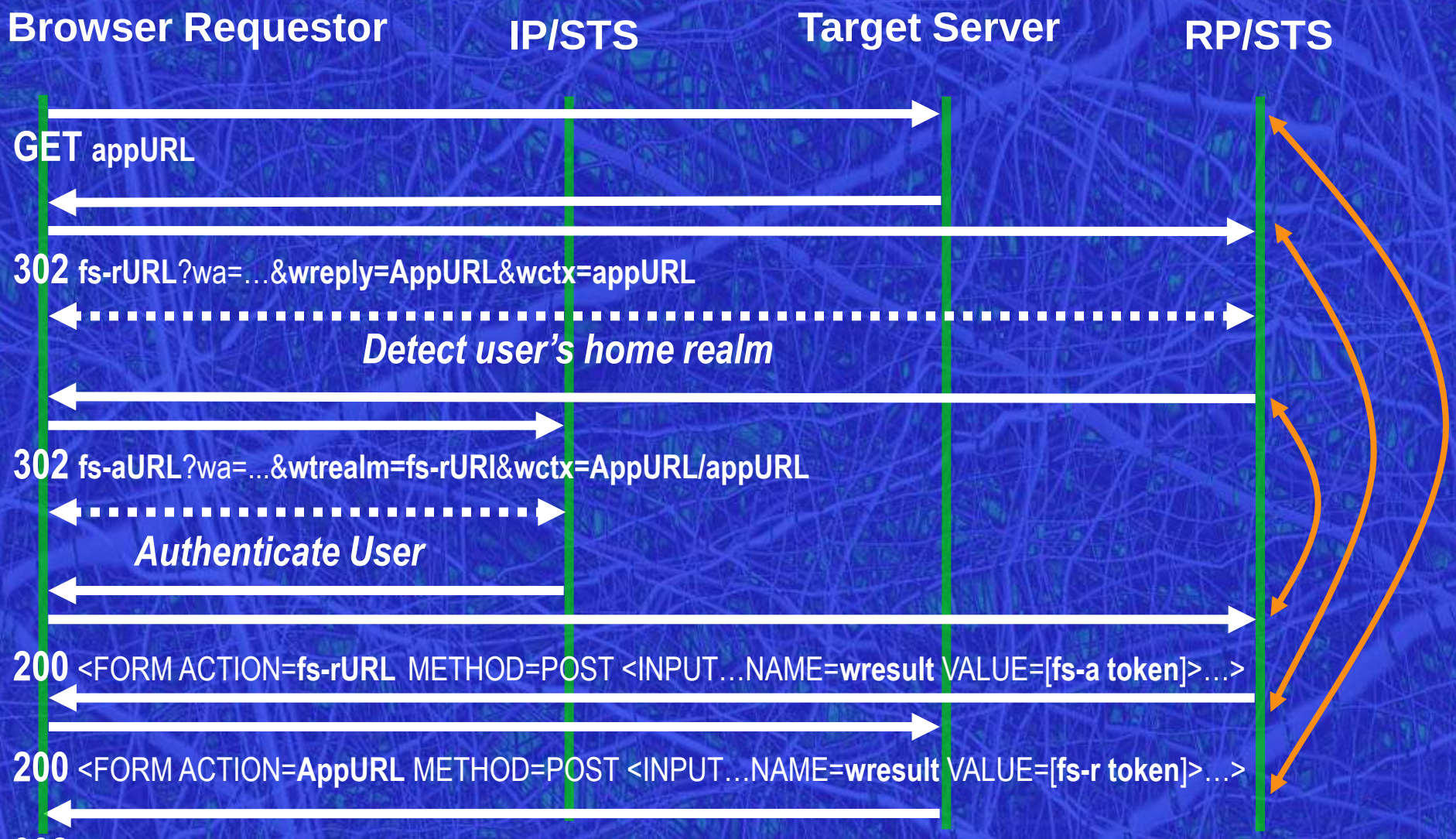
- Mappings defined for parameters to RST parameters
- A “ctx” parameter is defined to save context between parties
- Parameters allow pointers (URLs) to RST and RSTR values allowing them to be pulled not pushed

SOAP Requestor Msg Flow

WS-SecurityPolicy drives request routing



Web Requestor Msg Flow



Web Requestors

Home Realm Discovery

- Different choices
  - Fixed
  - Based on Requestor IP
  - Passed in
  - Prompt
  - Discovery service
    - Redirection through service
    - Allows for service-specific discovery proccess
    - Result returned in separate parameter
  - Shared cookie (not covered)

Agenda Part 6

Summary

- Goals & Requirements recap

Goals and Requirements Recap

- Promote identity federation
  - Enhance WS-Trust STS support for distributed authentication and authorization across realm boundaries
  - Make identity mapping optional (for privacy or personalization)
  - Enable different levels of privacy for different types of personally identifying information
- WS-Federation coverage
  - Section 2. Federation Model
  - Section 8. Additional WS-Trust Extensions
  - Section 12. Privacy

Goals and Requirements Recap

- Reduce operational friction in federations
  - Support mix & match of trust topologies and token types
  - Enable automated configuration using Federation Metadata
  - Allow single infrastructure to serve both SOAP and Web requesters
- WS-Federation coverage
  - Section 2. Federation Model
  - Section 3. Federation Metadata
  - Section 10. Indicating Specific Policy/Metadata
  - Section 4. Sign-Out
  - Section 13. Web (Passive) Requestors

Goals and Requirements Recap

- Reuse the WS-Trust STS model
  - Offer common interface for broad range of federation services
  - Allow identity, authentication, and authorization data to be shared as claims without requiring a specific token type
- WS-Federation coverage
  - Section 2. Federation Model
  - Section 5. Attribute Service
  - Section 6. Pseudonym Service
  - Section 7. Security Tokens and Pseudonyms
  - Section 9. Authorization