



The Cougar Component Model: or How I Learned to Stop Worrying

Michael Thome (mthome@bbn.com)

Todd Wright (twright@bbn.com)

Sebastien Rosset (srosset@cougaarsoftware.com)

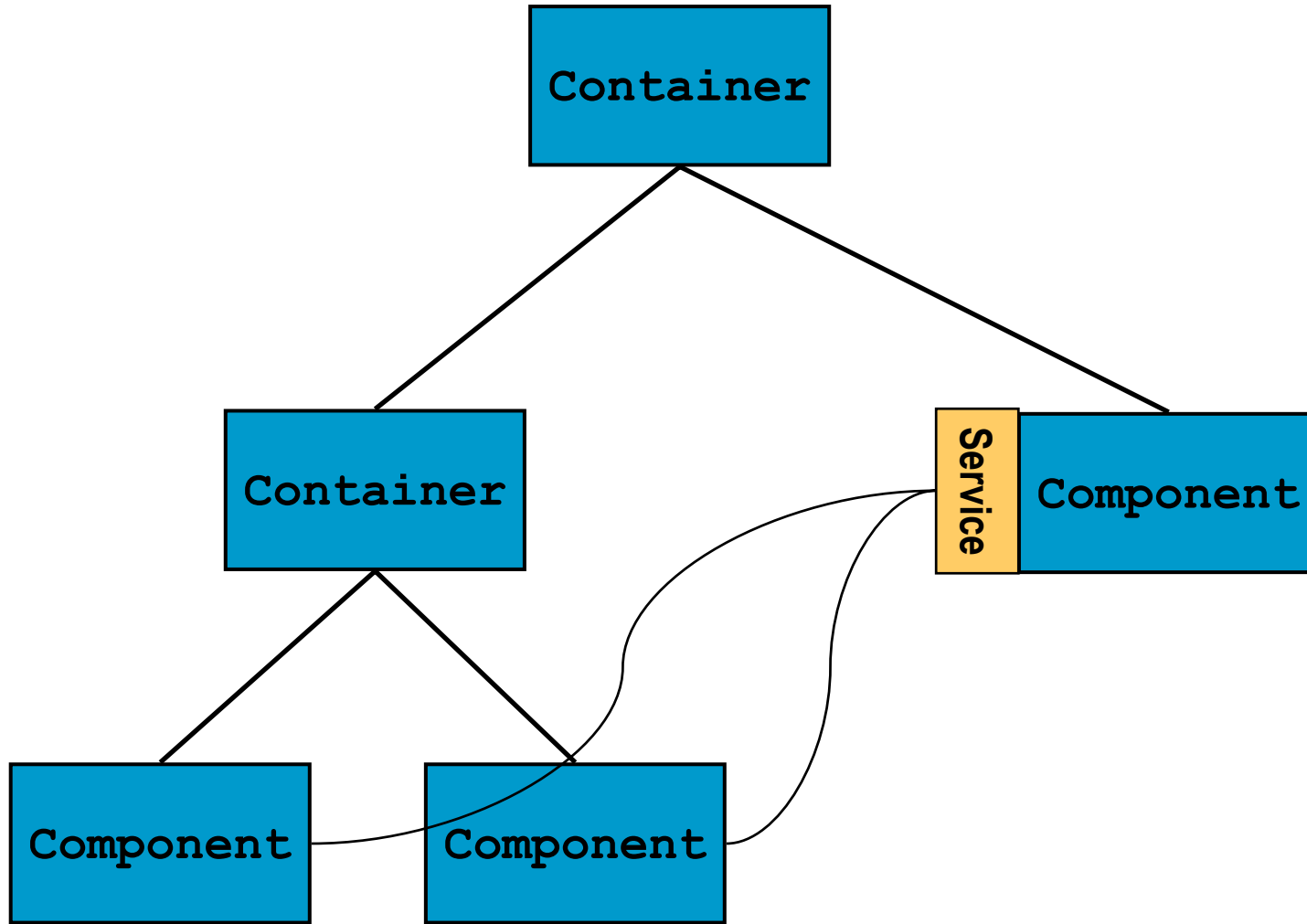


- What is a Component Model?
- What is different about Cougaar's Component Model?
- In what ways can one use the Cougaar Component Model?
- Case Study: UltraLog Security Services

- Framework for loading modular software
- Adds application runtime relationship maintenance to Object Model
- Service model
 - Interface enforcement
 - Dynamic discovery
- See: Java Beans

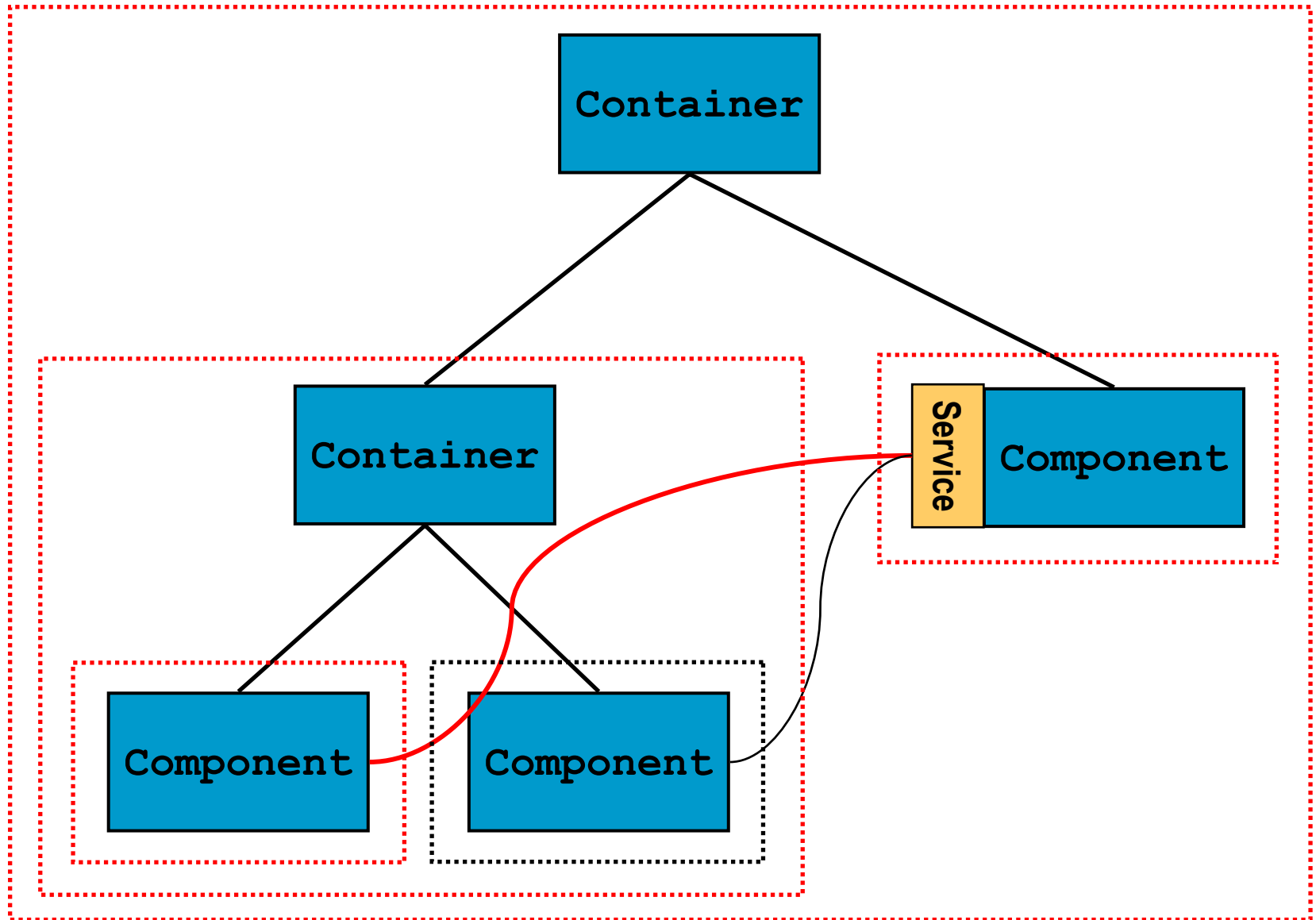
- Separate Containment from Service
- Dynamic component lifecycle
- Description-based specification
- Inter-component relationships defined by:
 - Position in component hierarchy
 - Services offered and requested

A Hierarchy of Components



- Active insulation surrounding each component
- Enforces strong encapsulation
- May audit, veto, proxy or transform all:
 - Component lifecycle events
 - Service hookups
 - Service invocations (inter-component method calls)
- Binders are themselves components

A "Bound" Hierarchy of Components





- The Cougaar infrastructure is fully “componentized”
- UltraLog survivability software



- Security services functional and CCM view
- Benefits of the security services componentization
- Security binders usage patterns
- Security binders examples
- Wiring of security components



■ **Secure Execution Environment**

- JAAS authentication
- JAR signature verification
- Java security manager

■ **Cryptographic services**

- Agent-based certificates authorities
- Public/private keys for users, agents, nodes and hosts
- Message encryption and signature
- Data encryption and signature
- Data recovery agents (key escrow)
- Certificate Revocation Lists
- HTTPs, SSL

■ **Hardened Environment**

- SELinux, snort, smartcards, tripwire
- Firewalls

■ **Policy services**

- Policy specification
- Conflict resolution
- Policy distribution

■ **Monitoring and Response services**

- Security console
- Event ontology
- Hierarchical managers
- Sensors

■ **Access Control Services**

- Message access control
- User authentication & authorization
- Community, naming service access control
- Blackboard access control

The UltraLog Security Services Cougaar Component Model View



Services	Data encryption and signature, policy service, authorization service, certificate cache, certificate directory, private key management, CRL management, LDAP certificate directory, agent trust, user management, SSL, agent identity, configuration parser
Service Providers	Providers for all services above
Binders	Message access control, Security service access control, WP/YP/community access control, event throttler, blackboard access control, JAAS component and agent secure context, secure threading service, audit
Components	Audit, message encryption and signature, security factories, policy bootstrapper, security sensors
Plugins	CA management, configuration management, monitoring and response, security sensors
Agents	Certificate authority, CRL manager, monitoring and response manager, policy manager, data recovery manager, configuration manager, user manager
Not CCM	Secure class loader, Tomcat authenticator, secure hook servlet, Java security manager

Componentization of the Security Services Benefits



- Security components are installed and configured based on threat analysis
- Scalability of developmental activities – functionality and access control can be implemented by separate teams
- Scalability of operational activities – threat assessment in large scale distributed systems will be fragmented
- Facilitates introduction of policies dynamically established over the lifetime of the system
- Separation between policy mediation and enforcement: allows replacement of policy enforcement components and policy management services
- Support for complex policies

- Access control mediation
 - Blackboard access control
 - User authorization
 - Message access control
- Service proxy
 - User and agent audit service
 - Message encryption, signature and signature verification
 - Principal-based authentication
 - QoS enforcement

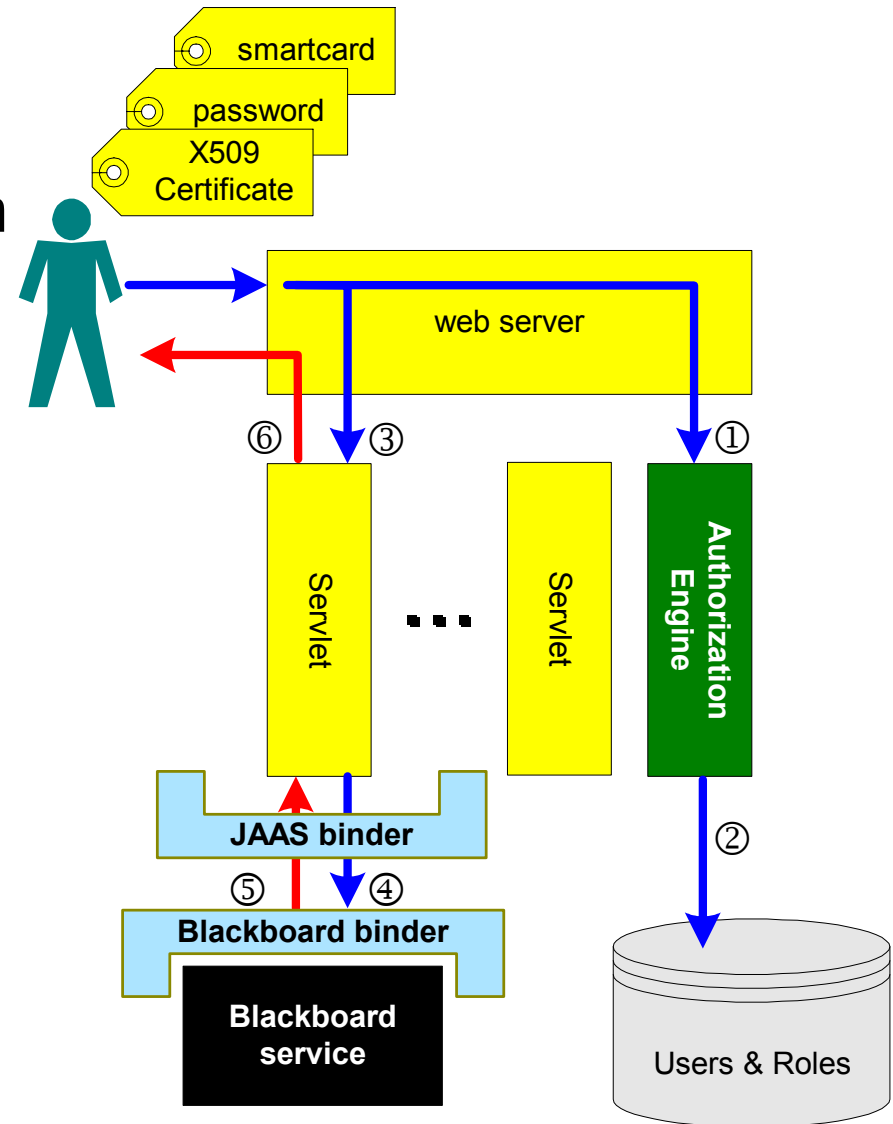


User Authentication and Authorization

Role-Based Access Control using security binders



- User invokes servlet
- User must provide credentials in accordance with security policy
- User is authenticated
- JAAS binder executes servlet in the security context of the authenticated user
- The servlet invokes the blackboard service
- A blackboard service proxy intercepts the request
 - Checks user role authorization

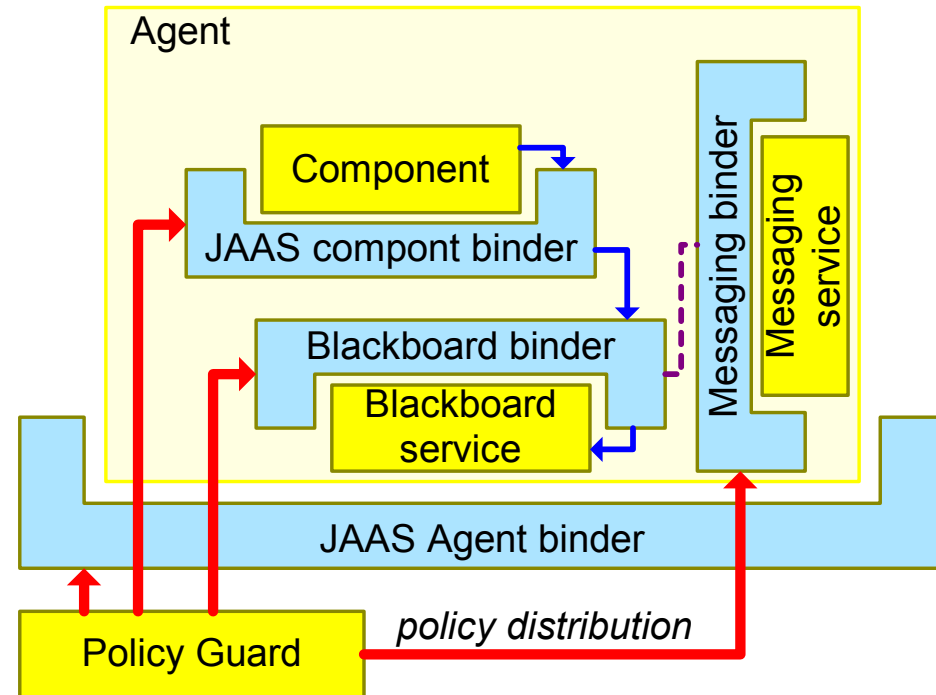


Agent and Component Identity

JAAS agent and component binders



- All components and agents run in their own security context
 - JAAS security context
 - Security context is used to enforce per-agent and per-component security policies
 - Accountability
 - Agent/Component identity is reported when there is a policy violation
 - Agent/Component identity can be reported when any exception occurs. Thread context includes Agent and Component identity
- Implemented as binders between the Agent Manager and the Agent

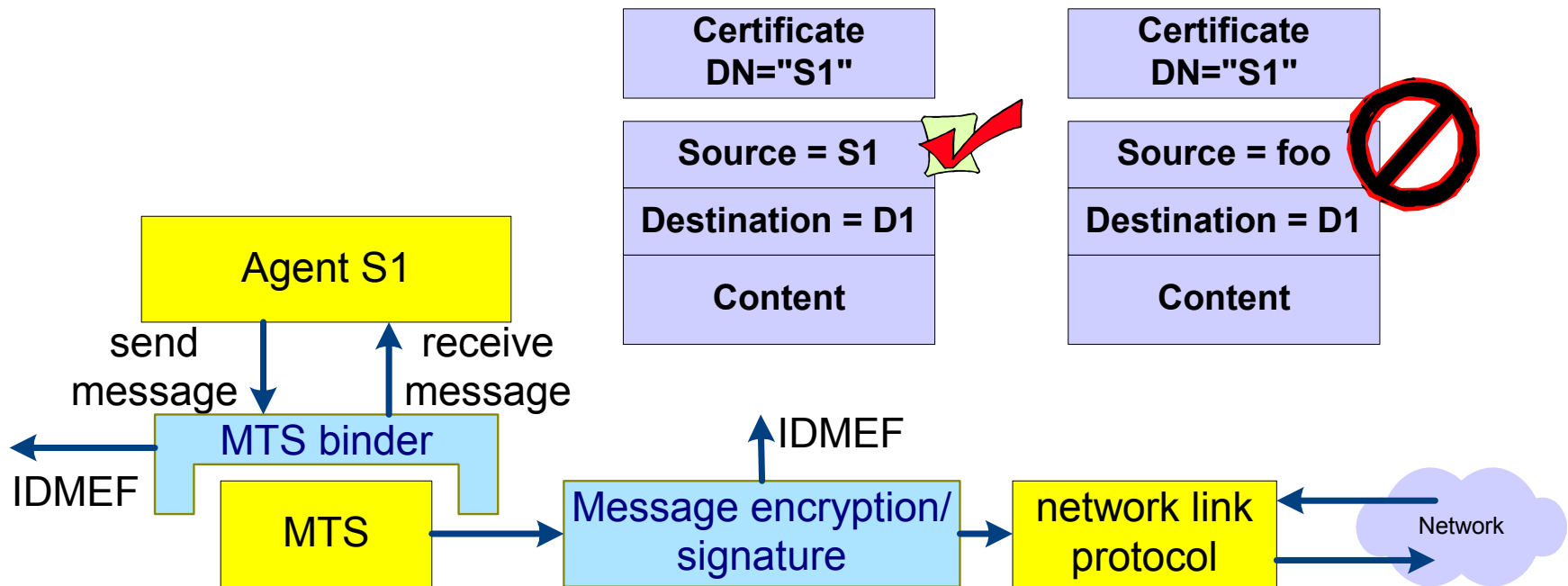


Message Access Control

Message Transport binder and message protection



- Protects agent
 - Accept/Reject incoming messages
- Controls agent
 - Accept/Reject outgoing messages
- Address validation
 - Verify outgoing messages have valid source address
- Policy-based access control
 - Access Control lists
 - Message tagging
- Generates IDMEF events when there is a policy violation

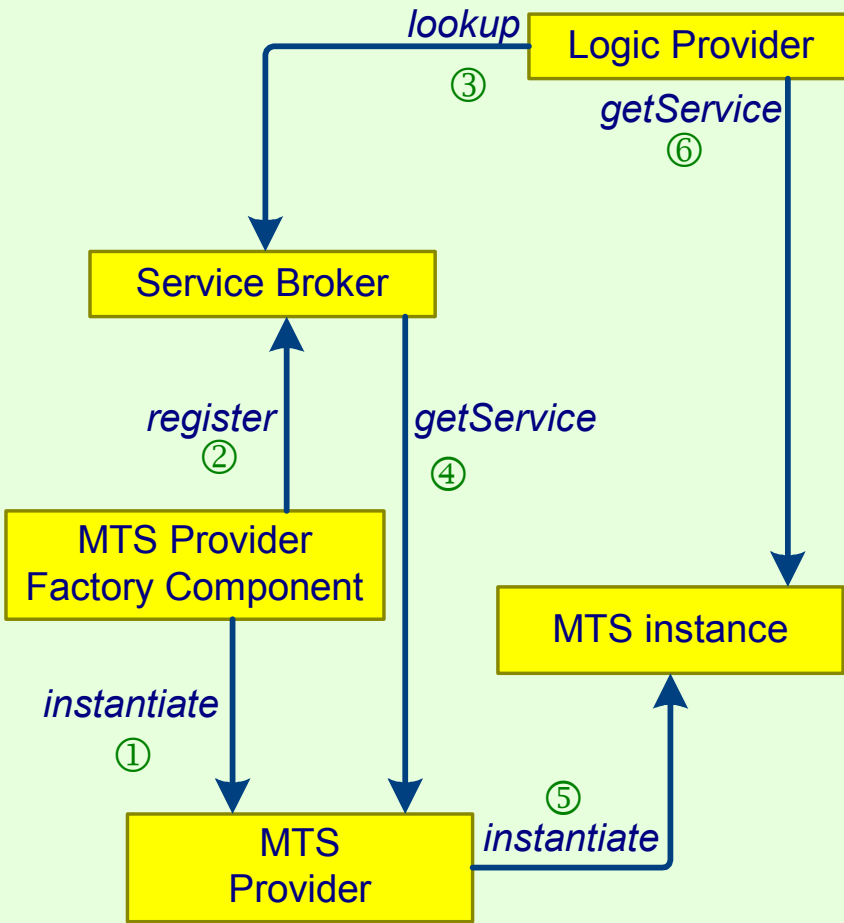


Security Binders

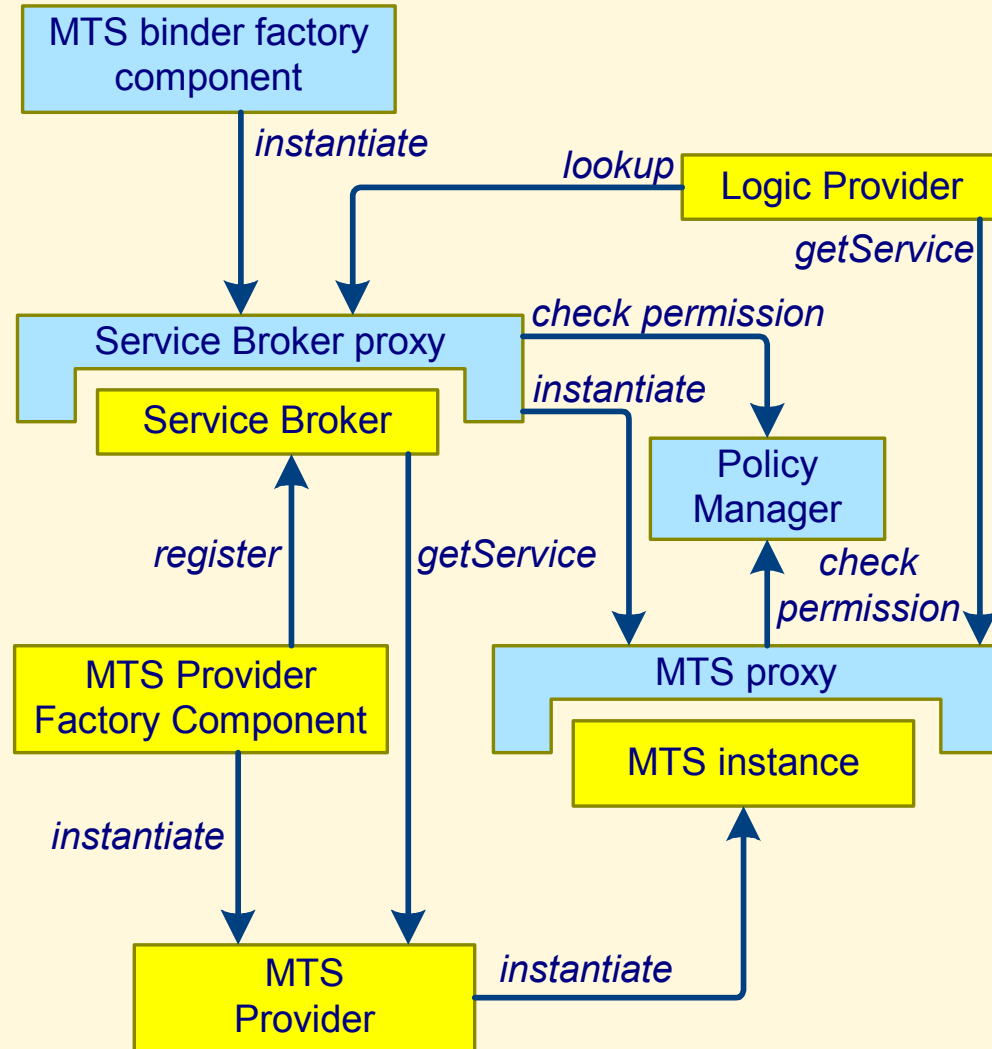
Wiring of Components



Without security



With security binders



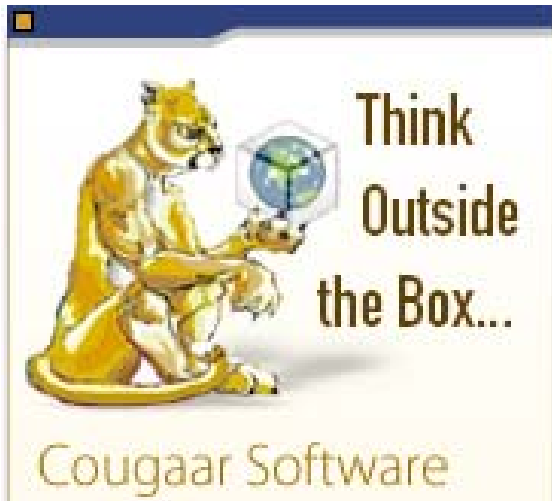
Questions?



<http://bbn.com>

Michael Thome (mthome@bbn.com)

Todd Wright (twright@bbn.com)



<http://cougaarsoftware.com>

Sebastien Rosset

(srosset@cougaarsoftware.com)