

OSGi Alliance Community Event

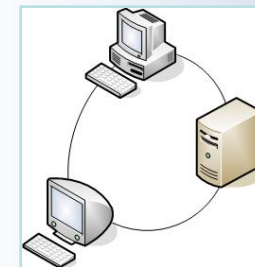
Distributed OSGi Services with the Eclipse Communication Framework

Jan S. Rellermeyer, ETH Zürich
Markus Kuppe, Versant Inc.



ECF: Eclipse Communication Framework

- Communication platform of Eclipse
 - Currently a Technology project
 - Will soon move to the Eclipse runtime project
- Goals
 - Support team and community collaboration
 - In combination with the Eclipse IDE
 - Shared editing, file transfer, messaging
 - As an interprocess communication platform for OSGi apps
 - E.g., service discovery, remote services

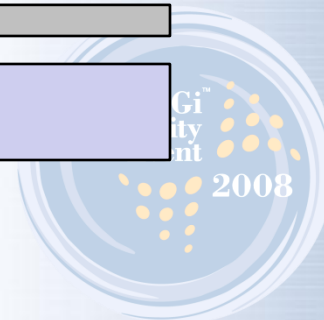
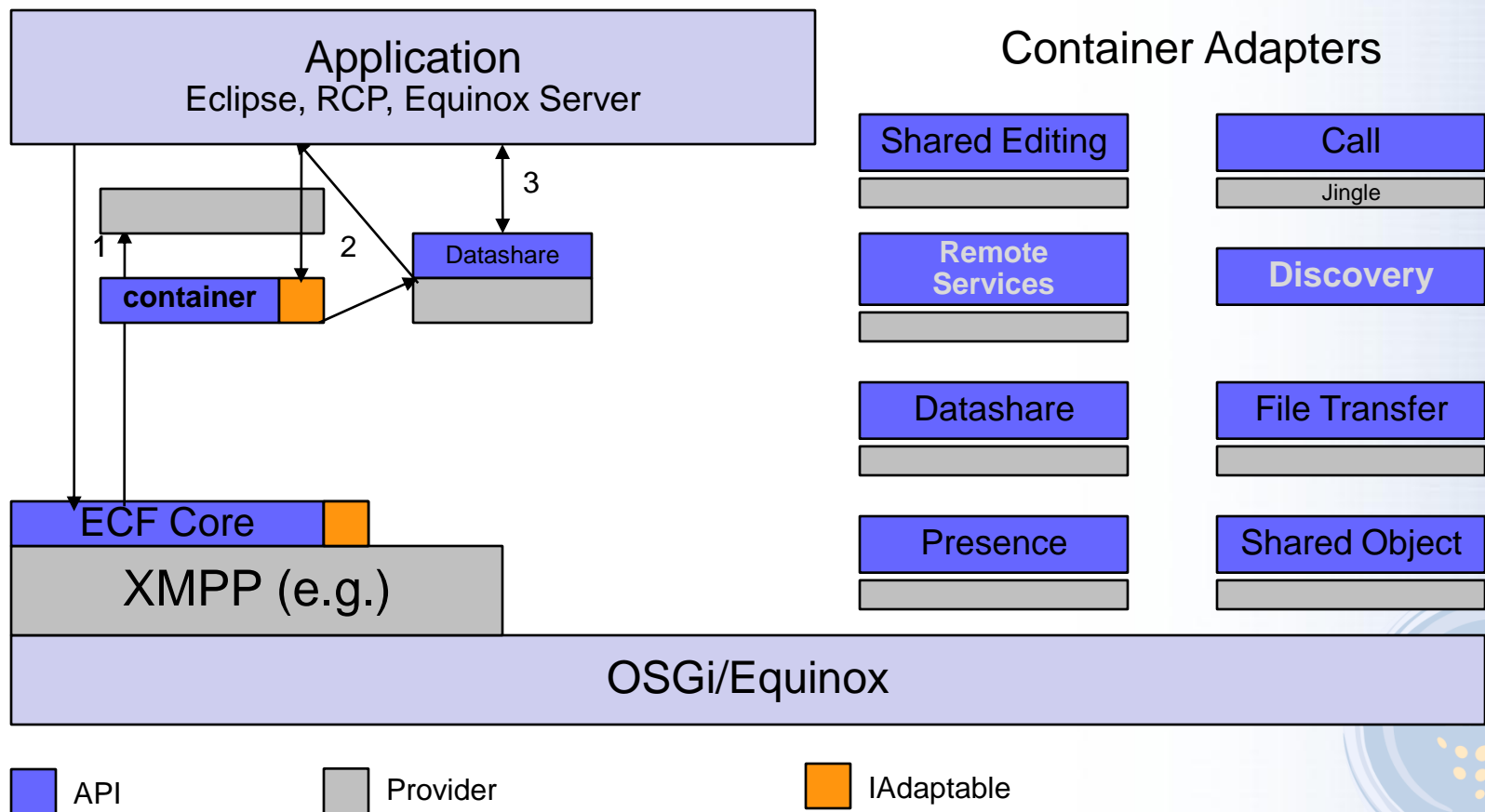


ECF adopters and applications

- Versant Corp. for Vitness
- Cloudsmith Inc. for Buckminster.
- Eclipse Platform includes Equinox p2
 - uses ECF for all network data transfer
 - will probably use ECF discovery in future versions
- IRC users use the KOS-MOS IRC bot (based on the ECF bot framework)
- you?

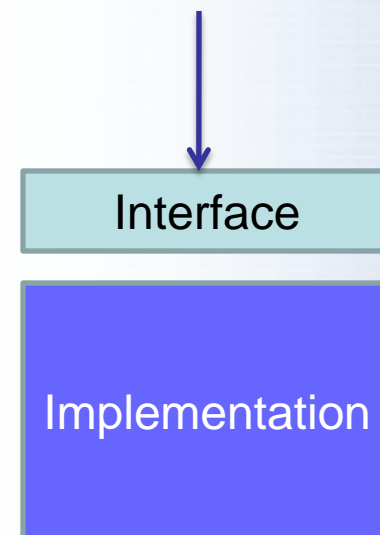


ECF Architecture



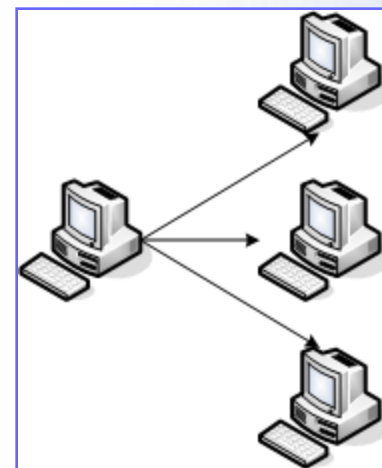
OSGi Services

- OSGi services provide
 - Encapsulation at a larger granularity
 - Loose coupling of functionality
 - Extensibility
 - Abstraction
- Remote services
 - Take this existing boundary to turn an application into a distributed application
 - Provide an abstraction to design distributed apps



OSGi services in the network

- Locate a service
 - Implementation for a given interface
 - Service discovery
 - Common knowledge
- Making use of a service
 - Providing service access via ECF API
 - “importing” the service into the local service registry
 - Providing a local service proxy



ECF service discovery – Overview

- Query for known/available services
 - Synchronous
 - Asynchronous: add/remove a service listener and get notified about service discovery/”undiscovery”
 - Query by filter/example (TODO)
- Manual and automatic service announcement



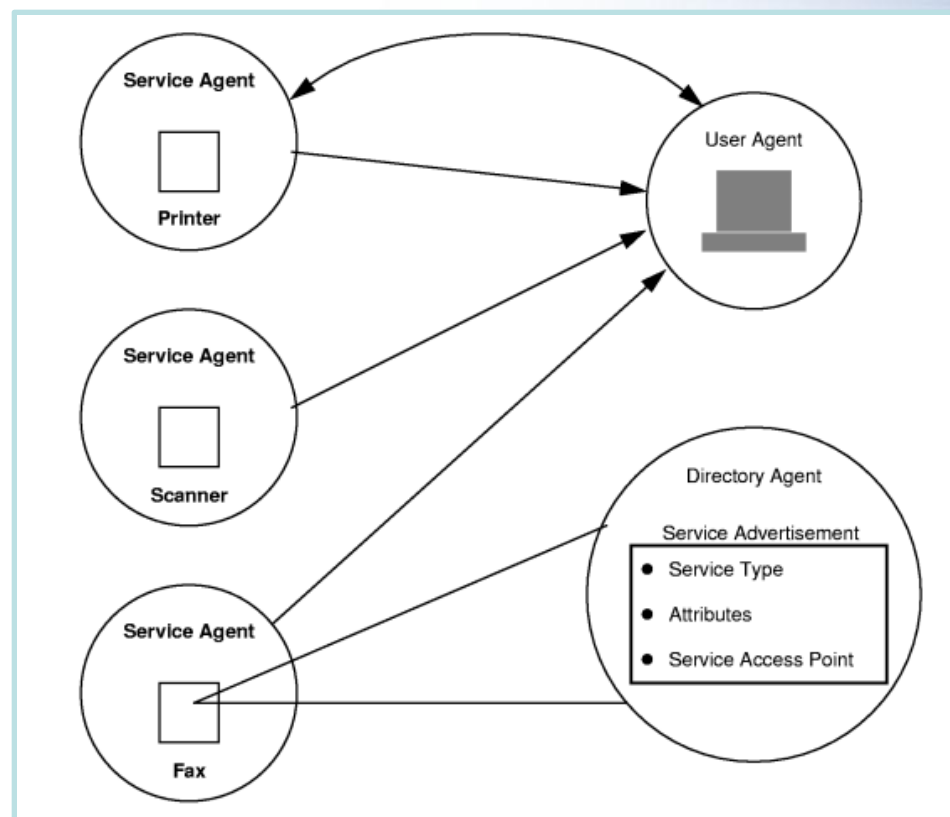
ECF discovery – Requirements

- Protocol and „space“ agnostic
 - Does not expose protocol internals
 - Not limited to, e.g., the LAN
 - Namespace/ID allows flexibility in service addressing
 - No strict borders to search
- Transparency
 - „automatic mode“
 - Announcement is just a service property in the OSGi context (Extender model if 3rd party bundle)
 - Listener is just a `org.osgi.framework.ServiceListener`
- Intransparency
 - „manual mode“
 - Consumer gets hold of discovery services and uses it



SLP Provider

- SLP protocol
 - Multicast discovery
 - Server (DA)
- Seamless transition
 - DA discovery
- Close(r) to OSGi Services
 - Service properties
 - LDAP filters



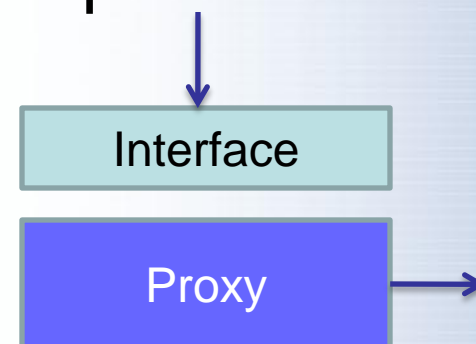
mDNS Provider

- DNS-SD on top of Multicast DNS
 - Multicast DNS: p2p name resolution
 - DNS-SD: service discovery
- Idea: Hosts are authoritative for their resources
- One shot and continuous queries
- Well-known from Zeroconf or Apple Bonjour



Remote Services

- OSGi services which cross address spaces
- Same ideas:
 - Ask for a service (-reference)
 - Can trigger service discovery
 - Get the service
 - Get a proxy for the service
 - Proxy generation can be proactive or reactive
 - Use the service
 - Method invocations become remote invocations



Transparent API

- Service and client remain untouched
- Some entity (not the client) states the demand
- Proxy is already present when the client asks for the service
- The service remains agnostic against distribution, as far as possible
- Seamless and flexible transition from local to remote services



Interface

Proxy



Non-Transparent API

- Client is aware of distribution
 - Retrieve an `IRemoteService` object
 - Explicit app-level failure handling
- Explicitly call remote invocations
- Call semantics can differ from local service calls
 - One-shot invocation (non-blocking)
 - Asynchronous invocation
 - E.g., with listener callback
 - Futures



Generic Provider

- DSO model
- Proactive, client/server model
 - In case of XMPP transport, the server is “hidden”
- Connected clients see all service proxies
- By default, no type injection
 - the assumption is that dependant types referenced by the service interfaces are known to all peers
 - Can be customized to go further
- Can be used with XMPP, JMS, JavaGroups



R-OSGi Provider

- R-OSGi was one of the first projects to enable remote OSGi services
- Is itself “just a service”
- Picks up services tagged for remote access
- Only the interface is transmitted
- Client builds a dynamic proxy
- Can be added to any OSGi runtime (R3 + R4)
- Protocol and transport-independent

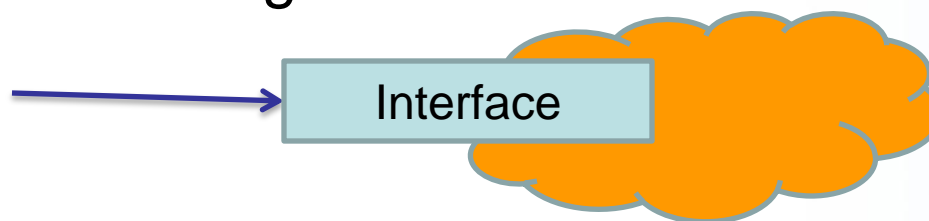


Interface



R-OSGi Proxy Bundles

- Synchronized lifecycle with the original bundle
 - Also involves the service properties
 - Changes are propagated to all proxies
- Self-contained units
 - Type Injection
 - Provides exactly the view on the bundle that a client has when looking at the service



What about RFC 119?

- Work in progress
- Will address service discovery and remote services
- More focus on integrating different concepts of distribution
 - Defines metadata for federating heterogeneous service concepts through OSGi
 - Scope includes even languages other than Java
- ECF as a reference implementation for 119?



Eclipse ECF Project

- <http://www.eclipse.org/ecf>
- <http://wiki.eclipse.org/ECF>
- ECF 2.0 will ship with Eclipse Ganymede
- The work on ECF 2.1 has just started 😊

Questions?

