# ECF on the Servers

# OR

# Equinox/OSGi + ECF = 'Equinox Service Bus'

***Scott Lewis -- slewis@composent.com***
***Mustafa Isik -- codesurgeon@gmail.com***
***Roland Fru -- roland@bitsvalley.com***

## ECF: What is it?

- Family of APIs for messaging
  - Very Small Core
    - Asynch messaging 'container'
    - extensibility mechanism
  - Extensibility with Adapters
    - data channels, object/model replication, service discovery, file transfer,presence/im/chat, voip call setup, remote services, others/your own...

# Overall Goals for Framework

- Usability
  - Minimalism:  Only use messaging APIs you need
  - API Consistency:  Consistent way of doing same thing (e.g. Filetransfer) over multiple protocols (e.g. Http, bittorrent)
- Interoperability, Integration
- Open protocols via Open source

## Now: ECF on Clients

- Eclipse-Based Tool Collaboration
  - Corona (tools collaboration)
  - ALF (process orchestration)
  - Integrated VOIP/IM/chat
  - Shared editing
  - Dev Team Collaboration (teams)
- RCP Apps/Tools
  - Any app that needs/wants communications/collaboration functionality

## Also Good for Server(s)

- Integration
  - Many protocols, one UI, one application 'model'
- Interoperability
    - Legacy systems/protocols via 'bridging' (OHF)
- Replication for load balancing
- Remote Services API for Server-Server Messaging

## ECF Remote Service API $^{\text{(new!)}}$

- Register service
  - IRemoteServiceContainer.registerService
- Service registry is **replicated within group** (using ECF provider)
- Intentionally separate from OSGi service registry

## ECF Remote Service API...on 'Client'

### Lookup

- `IRemoteServiceReference[] refs = getRemoteServiceReferences(...);`
- `IRemoteService rservice = getRemoteService(refs[0]);`
- THEN, 4 explicit options for remote service invocation
  - `1. Object proxy = IRemoteService.getProxy()`
    - Call/return. Blocks until result
  - `2. Object result = IRemoteService.callSynch(IRemoteCall)`
    - Call/return. Blocks until result
  - `3. IRemoteService.callAsynch(IRemoteCall, listener)`
    - Call/return. No blocking (listener notified)
  - `4. IRemoteService.fireAsynch(IRemoteCall)`
    - 'Fire and go'. No block (no success/failure info)

## ECF Remote Service API: Summary

- Looks very similar to OSGi services
  - BundleContext.registerService
  - BundleContext.getServiceReferences(...)
  - BundleContext.getService(ref)
- Separate but Equal
- Transport can be: JMS, ECF generic, XMPP, RMI, XML-RPC, SOAP, others

## Server-Side ECF='Equinox Message Bus'

- Protocols dynamically added/loaded
  - Server can talk different protocols to get same API/semantics
- Communications components/bundles can consistently be created for
  - Eclipse plugins
  - RCP Apps
  - Servers

## Conclusion

- Website
  - http://www.eclipse.org/ecf
- Mailing List
  - http://dev.eclipse.org/mailman/listinfo/ecf-dev
- Newsgroup
  - news://news.eclipse.org/eclipse.technology.ecf