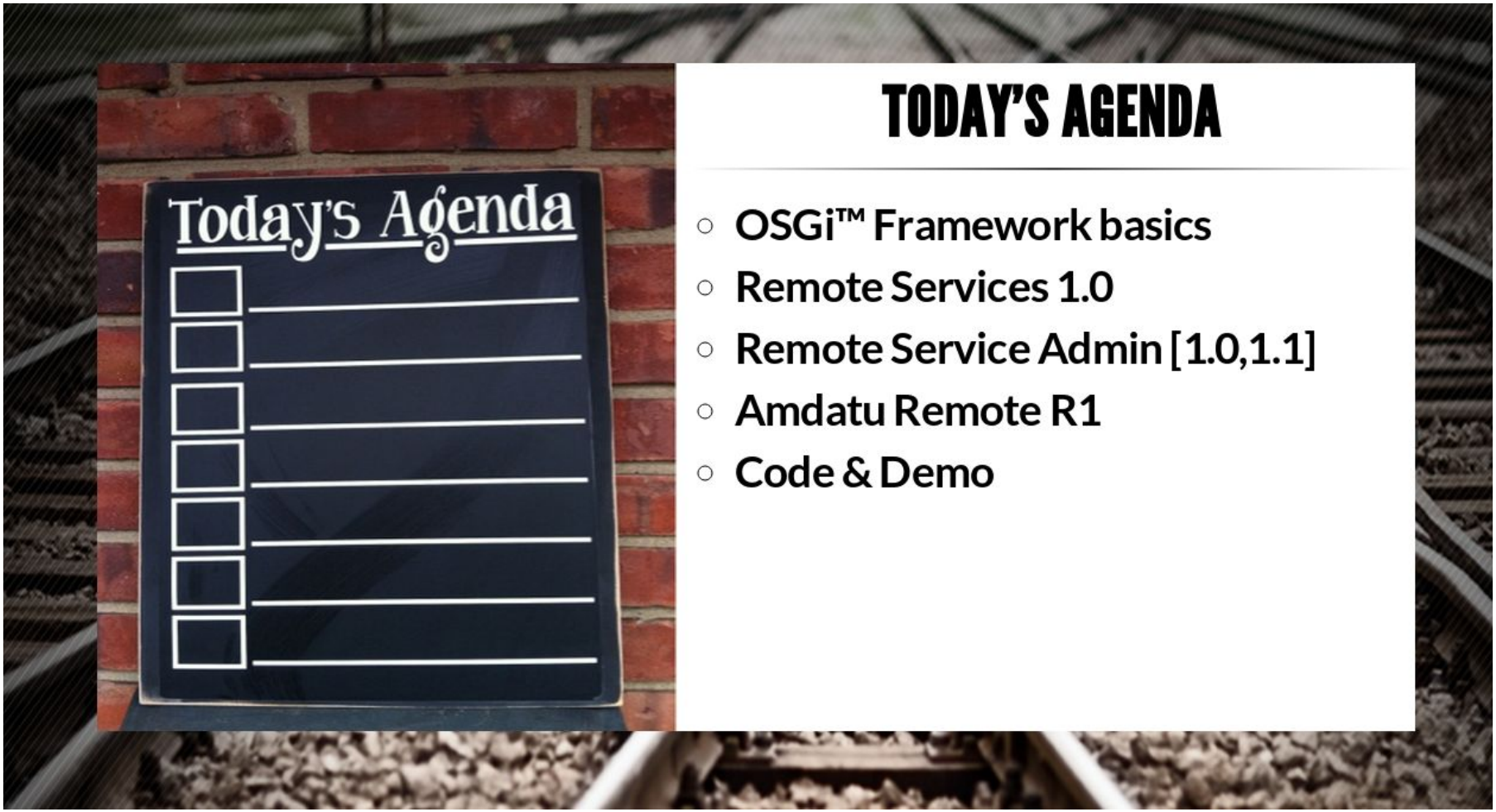

amdatu
AMDATU REMOTE
REMOTE SERVICE ADMIN 1.1 UPDATE


luminis
Conversing worlds
Bram de Kruijff
Luminis Technologies
@bdekruijff | @luministech



TODAY'S AGENDA

- OSGi™ Framework basics
- Remote Services 1.0
- Remote Service Admin [1.0,1.1]
- Amdatu Remote R1
- Code & Demo

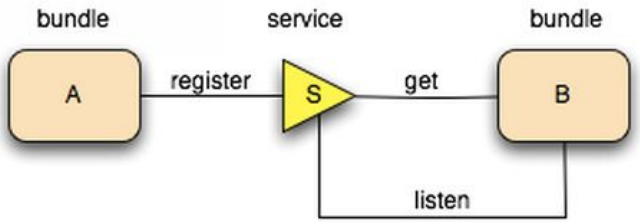
OSGI™ FRAMEWORK

- Specified by OSGi™ Alliance
  - Modularity to Java™
  - Dynamic services (in-VM)
- OSS implementations
  - Apache™ Felix
  - Eclipse™ Equinox
  - Knopflerfish™

MODULE LAYER

- Unit of modularization:
  Bundle
  - Packaging -> Java™ Archive
  - Metadata -> Manifest
- Sharing of packages
  - Strict versioning
  - Visibility enforced
- Dynamic Lifecycle
  - Activator hook

SERVICE LAYER



- Framework Service Registry
  - Dynamic (de)registration
  - Dynamic lookup/query
  - Event Listeners
- Service instances are POJO's
- Lookup based on filters (LDAP)

SERVICE REGISTRATION EXAMPLE

```
public class WorkerRegistrar {

    private ServiceRegistration<Worker> registration;

    public void register(BundleContext context) throws Exception {

        Dictionary<String, Object> properties = new Hashtable<>();
        properties.put("demo.worker.id", "worker1");

        registration = context.registerService(Worker.class, new WorkerImpl(), properties);
    }

    public void unregister(BundleContext context) throws Exception {
        registration.unregister();
    }
}
```

SERVICE LOCATION/INVOCATION EXAMPLE

```
public class WorkerCaller {

    public boolean call(BundleContext context) throws Exception {

        Collection<ServiceReference<Worker>> references =
           context.getServiceReferences(Worker.class, "(demo.worker.id=worker1)");

        if (!references.isEmpty()) {

            ServiceReference<Worker> reference = references.iterator().next();
            Worker worker = context.getService(reference);
            if (producer != null) {
                producer.getWork();
                context.ungetService(reference);
                return true;
            }
        }
        return false;
    }
}
```

SERVICE LISTENING EXAMPLE

```
public class WorkerListener implements ServiceListener {

    public void register(BundleContext context) throws Exception {
        context.addServiceListener(this, "(&(objectClass=demo.Worker)(demo.worker.id=worker1))");
    }

    @Override
    public void serviceChanged(ServiceEvent event) {

        switch (event.getType()) {
        case ServiceEvent.REGISTERED:
            // Do something
            break;
        default:
            break;
        }
    }
}
```

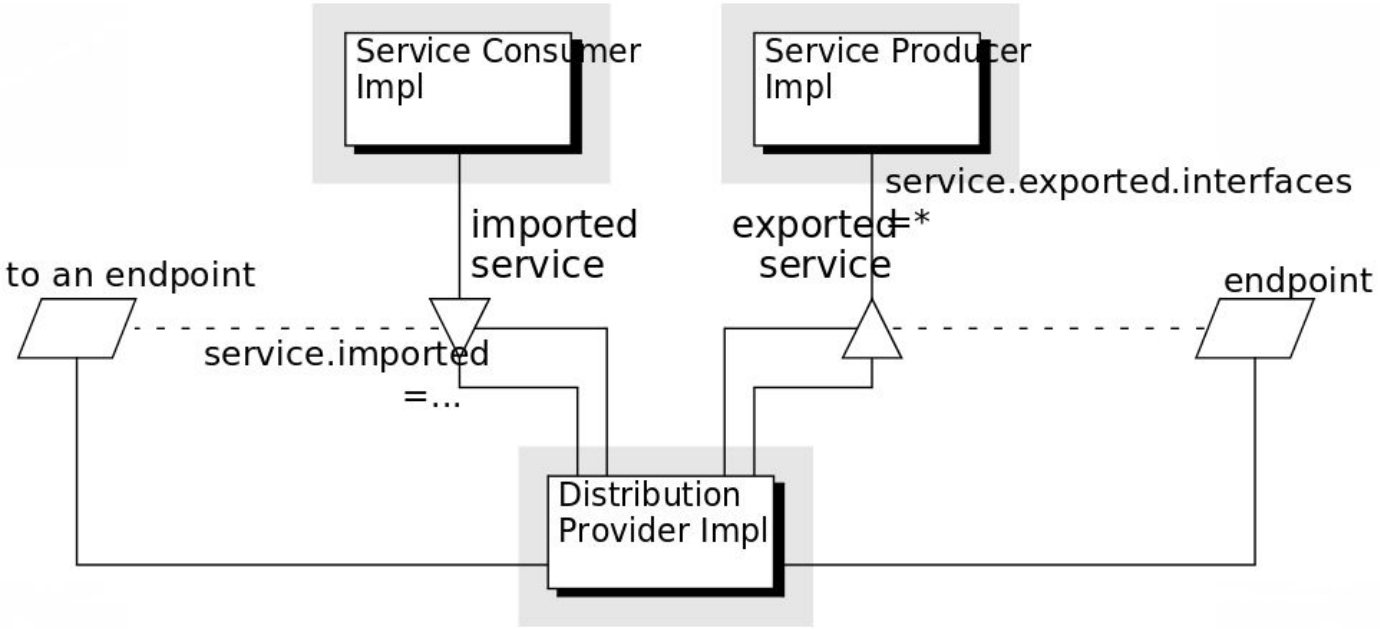
DEPENDENCY MANAGEMENT

- Managed availibility
- Dependency injection
- Reduced boilerplate
- DM frameworks
  - Declarative Services
  - Apache™ Felix DM

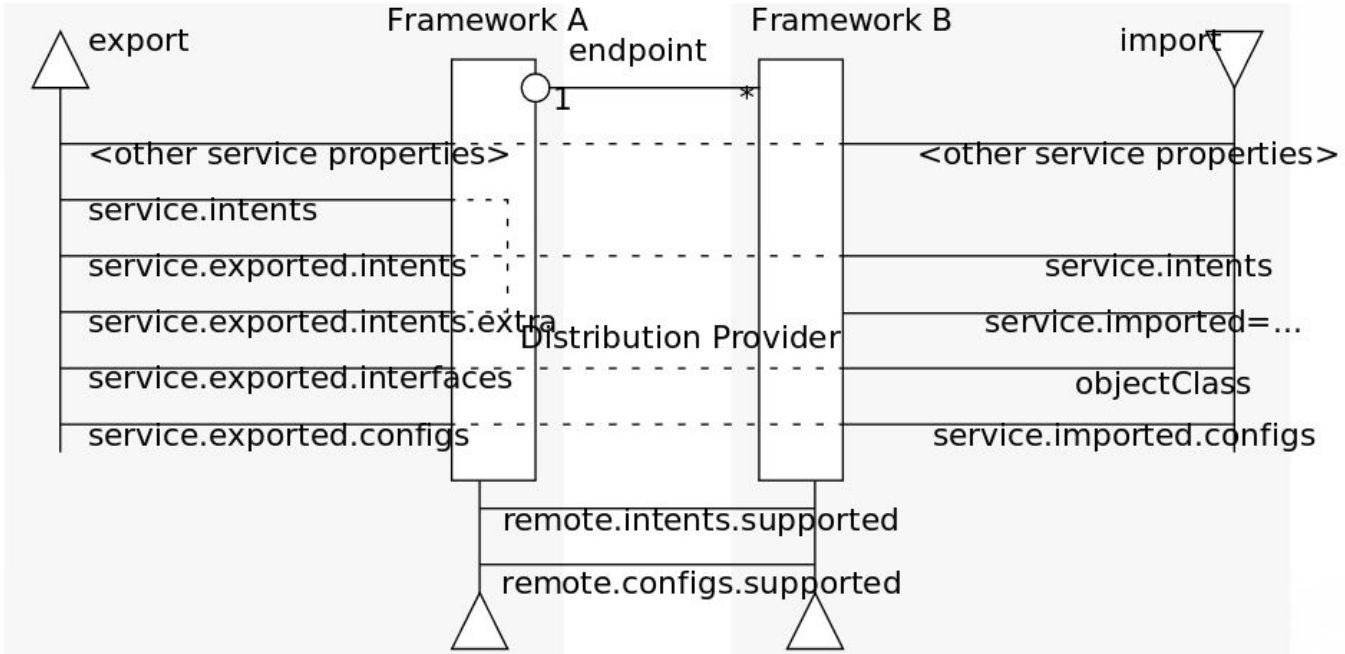
REMOTE SERVICES 1.0

- Since OSGi cmpn 4.2
- Transparent remoting
- Mechanical model
  - Distribution Provider
  - Configuration Types
  - Intents

REMOTE SERVICES ARCHITECTURE



REMOTE SERVICES PROPERTIES



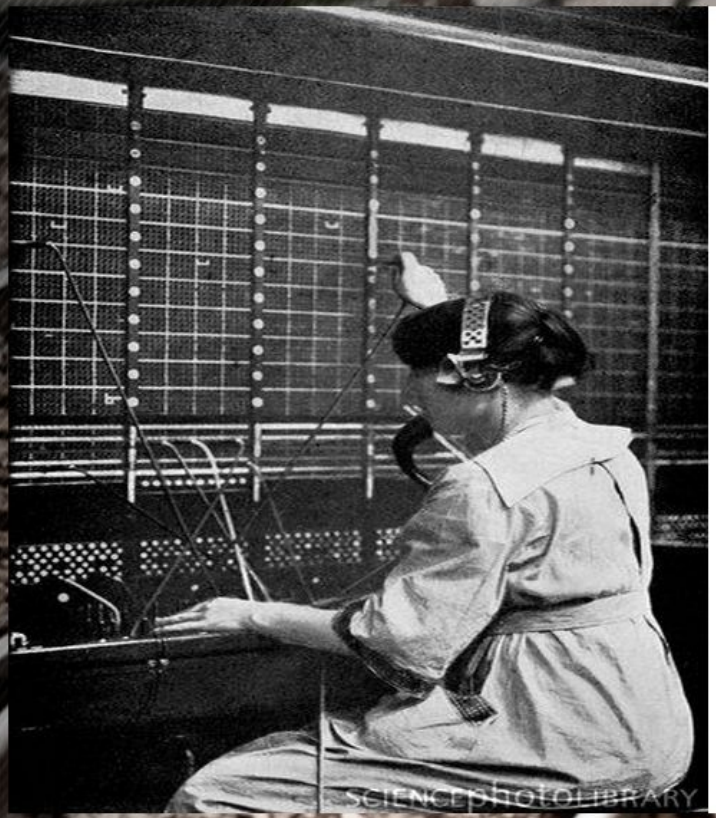
EXPORTING A SERVICE EXAMPLE

```
public class WorkerServiceExporter {

    private ServiceRegistration<Worker> registration;

    public void register(BundleContext context) throws Exception {

        Dictionary<String, Object> properties = new Hashtable<>();
        properties.put("service.exported.interfaces", Worker.class.getName());
        properties.put("demo.worker.id", "worker1");

        registration = context.registerService(Worker.class, new WorkerImpl(), properties);
    }

    public void unregister(BundleContext context) throws Exception {
        registration.unregister();
    }
}
```

RS EVALUATED

- ✓ Scale-out
- ✓ Transparent
- ✓ Pluggable

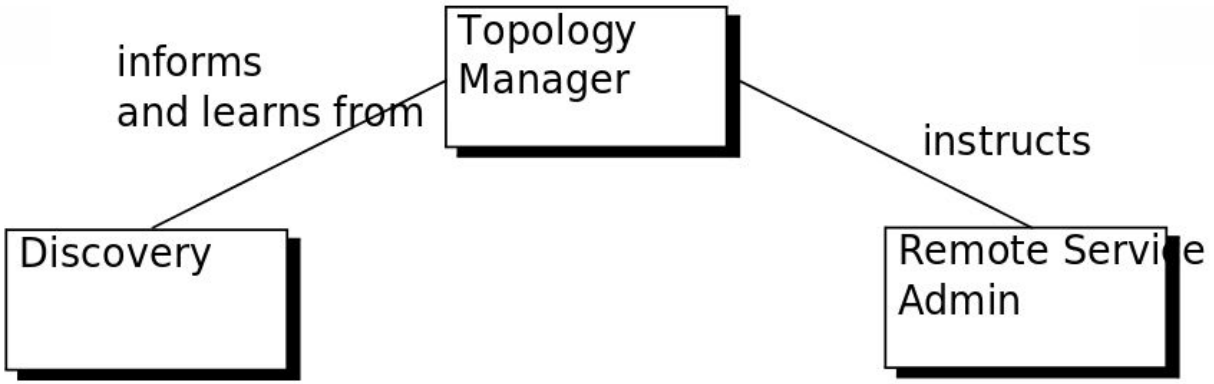
- ✗ Manageables
- ✗ Reusable

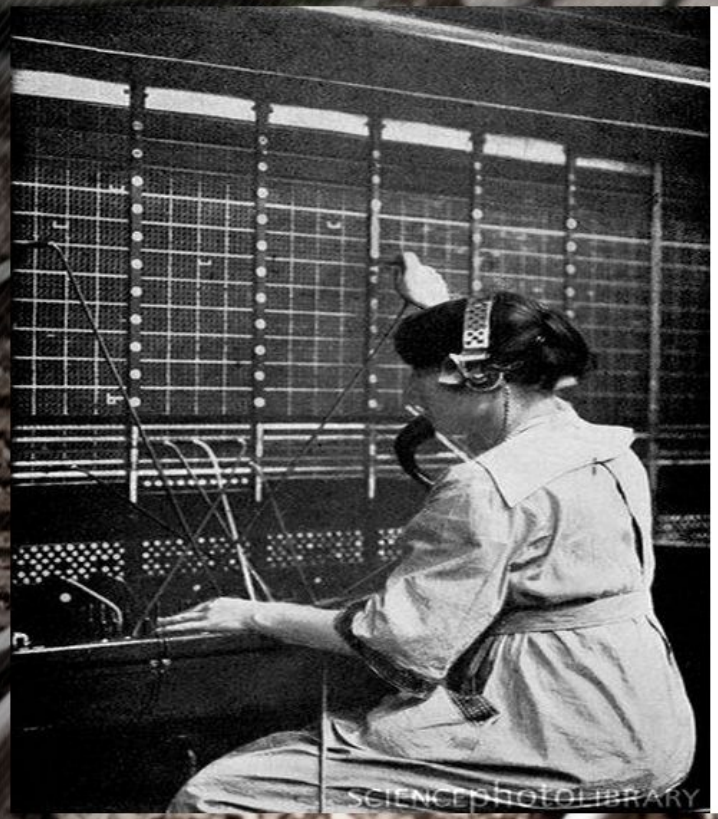


REMOTE SERVICE ADMIN [1.0,1.1]

- Since OSGi Enterprise 5
- Update OSGi Enterprise 6
- Decoupled responsibilities
  - Distribution Provider
  - Discovery Provider
  - Topology Manager

REMOTE SERVICE ADMIN ROLES

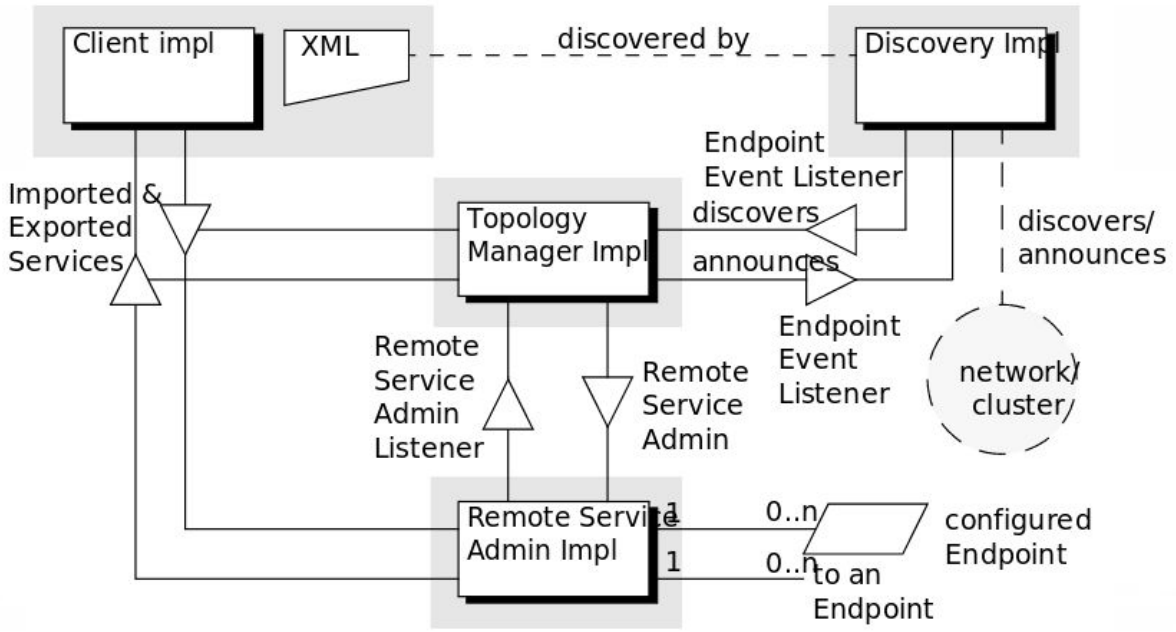




TOPOLOGY MANAGEMENT

- Basic
  - Promiscuous
  - Failover
- Coordination
  - Load Balancing
  - Content Based Routing
  - Policy driven

REMOTE SERVICE ADMIN ARCHITECTURE



ENDPOINT LISTENERS

- EndpointListener
  @deprecated 1.1
- EndpointEventListener
  added in 1.1
- Late joiners informed
- Filter based scoping

```
public interface EndpointListener {

 String ENDPOINT_LISTENER_SCOPE =
  "endpoint.listener.scope";

 void endpointAdded(
  EndpointDescription endpoint, String filter

 void endpointRemoved(
  EndpointDescription endpoint, String filter
}
```

```
public interface EndpointEventListener {

  String ENDPOINT_LISTENER_SCOPE =
         "endpoint.listener.scope";

 void endpointChanged(EndpointEvent event,
         String filter);
}
```

REMOTE SERVICE ADMIN

- Passive component
- Supports n config types
- Informs listeners
- Introspection methods
- ServiceFactory pattern provides auto cleanup

```
public interface RemoteServiceAdmin {

 Collection<ExportRegistration> exportService
  ServiceReference<?> reference,
  Map<String, ?> properties);

 ImportRegistration importService(
  EndpointDescription endpoint);

 Collection<ExportReference>
  getExportedServices();

 Collection<ImportReference>
  getImportedEndpoints();
}
```

```
public interface RemoteServiceAdminListener {

 void remoteAdminEvent(
  RemoteServiceAdminEvent event);
}
```

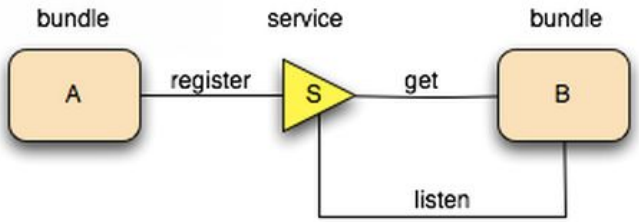
IM/EXPORT REGISTRATIONS

- Registration handles for imports & exports
- Update support allows propagation of modified registration properties @since 1.1

```
public interface ExportRegistration {

 ExportReference getExportReference();

 EndpointDescription update(
  Map<String, ?> props);

 void close();
 Throwable getException();
}
```

```
public interface ImportRegistration {

 ImportReference getImportReference();

 boolean update(EndpointDescription endpoint)

 void close();
 Throwable getException();
}
```

IMPORTED ENDPOINTS



- Java™ Proxies exposed as standard services
- ServiceFactory pattern ensures consumer bundle assignability

CONFIGURATION EXTENDER SUPPORT

Extender support for Endpoint Descriptions provided as an XML resources

```
Remote-Service: /resources/endpoints.xml
```

```
<endpoint-descriptions xmlns="http://www.osgi.org/xmlns/rsa/v.1.0.0">
 <endpoint-description>
  <property name="endpoint.framework.uuid" value="..."></property>
  <property name="endpoint.id" value="..."></property>
  <property name="endpoint.service.id" value-type="Long" value="..."></property>
  <property name="objectClass">
    <array><value>demo.Worker</value></array>
  </property>
  ...
 </endpoint-description>
</endpoint-descriptions>
```

CAPABILITY NAMESPACES

- Allow modules to express capability requirements
- Namespaces @since 1.1
  - Discovery Extender
  - Discovery Provider
  - Distribution Provider
  - Discovery Provider

capability:

```
Provide-Capability: \
  osgi.extender; \
  osgi.extender=osgi.remoteserviceadmin; \
  version:Version=1.1; \
  uses:="org.osgi.service.remoteserviceadmin"
```

requirement

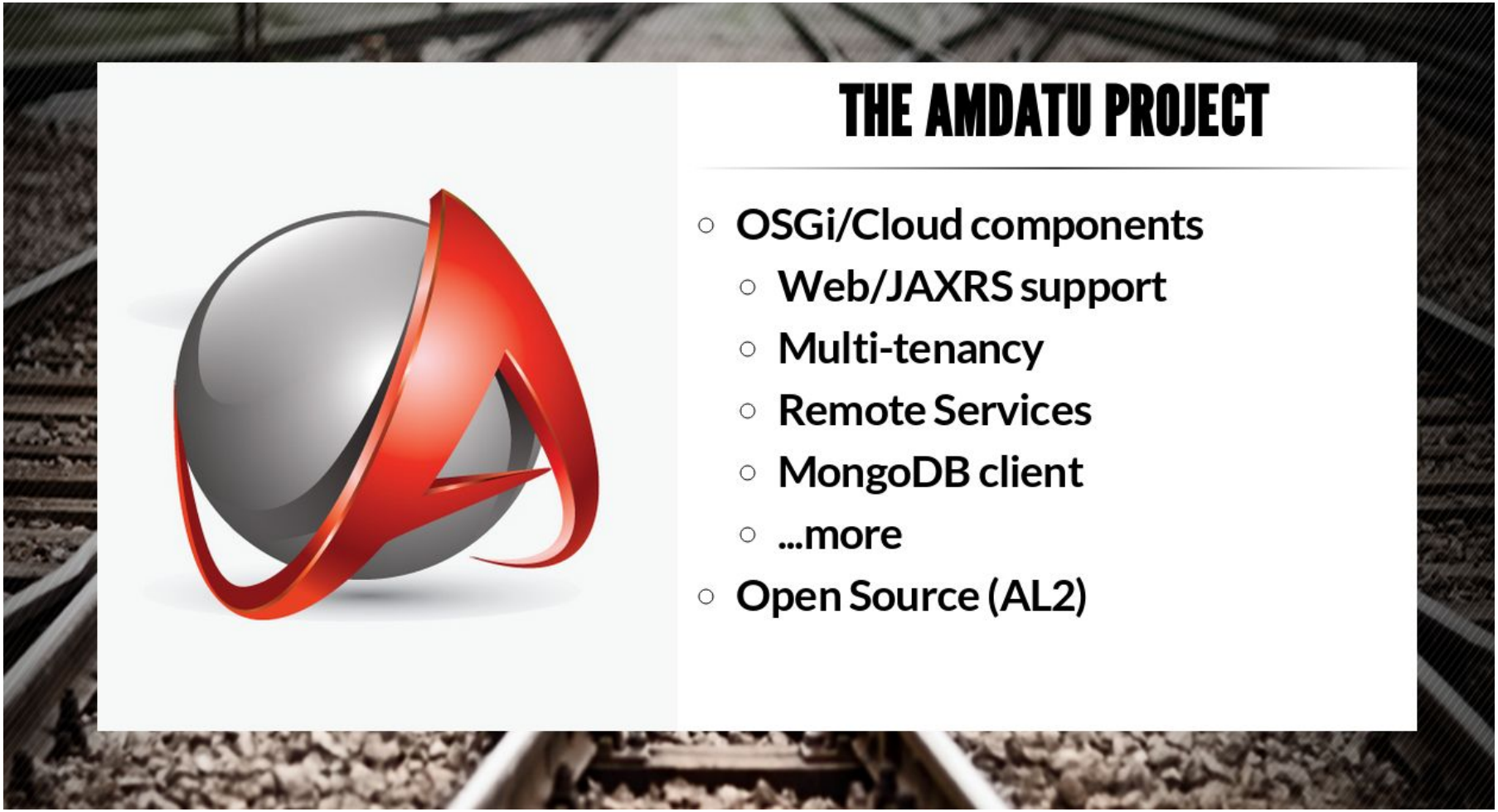
```
Require-Capability: \
  osgi.extender; \
  filter:=\
   (&(osgi.extender=osgi.remoteserviceadmin)
   (version>=1.1)(!(version>=2.0));
```

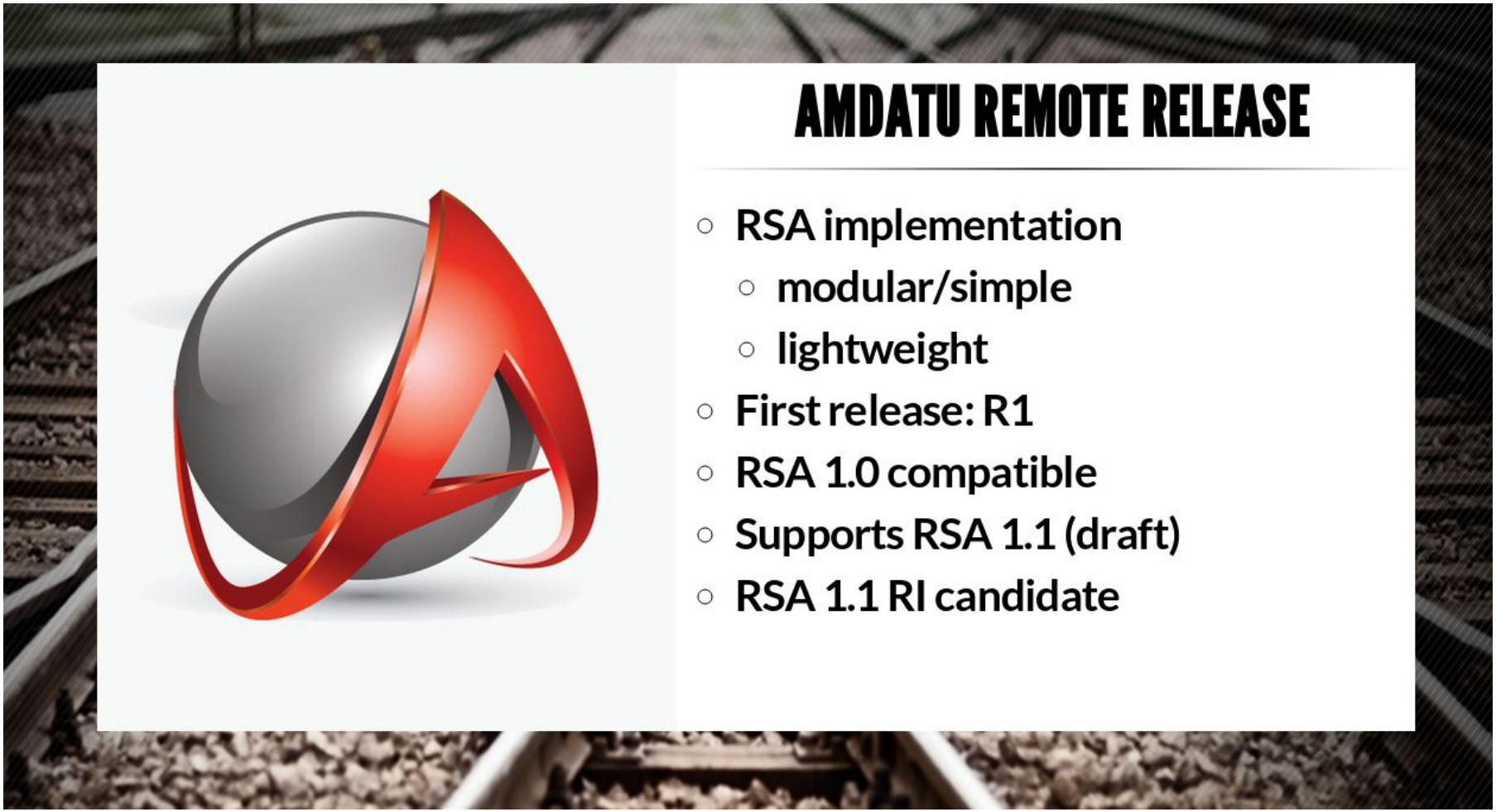
ADDITIONAL @SINCE 1.1

- Endpoint ID uniqueness clarification
- Removed Endpoint sharing requirement

RSA EVALUATED

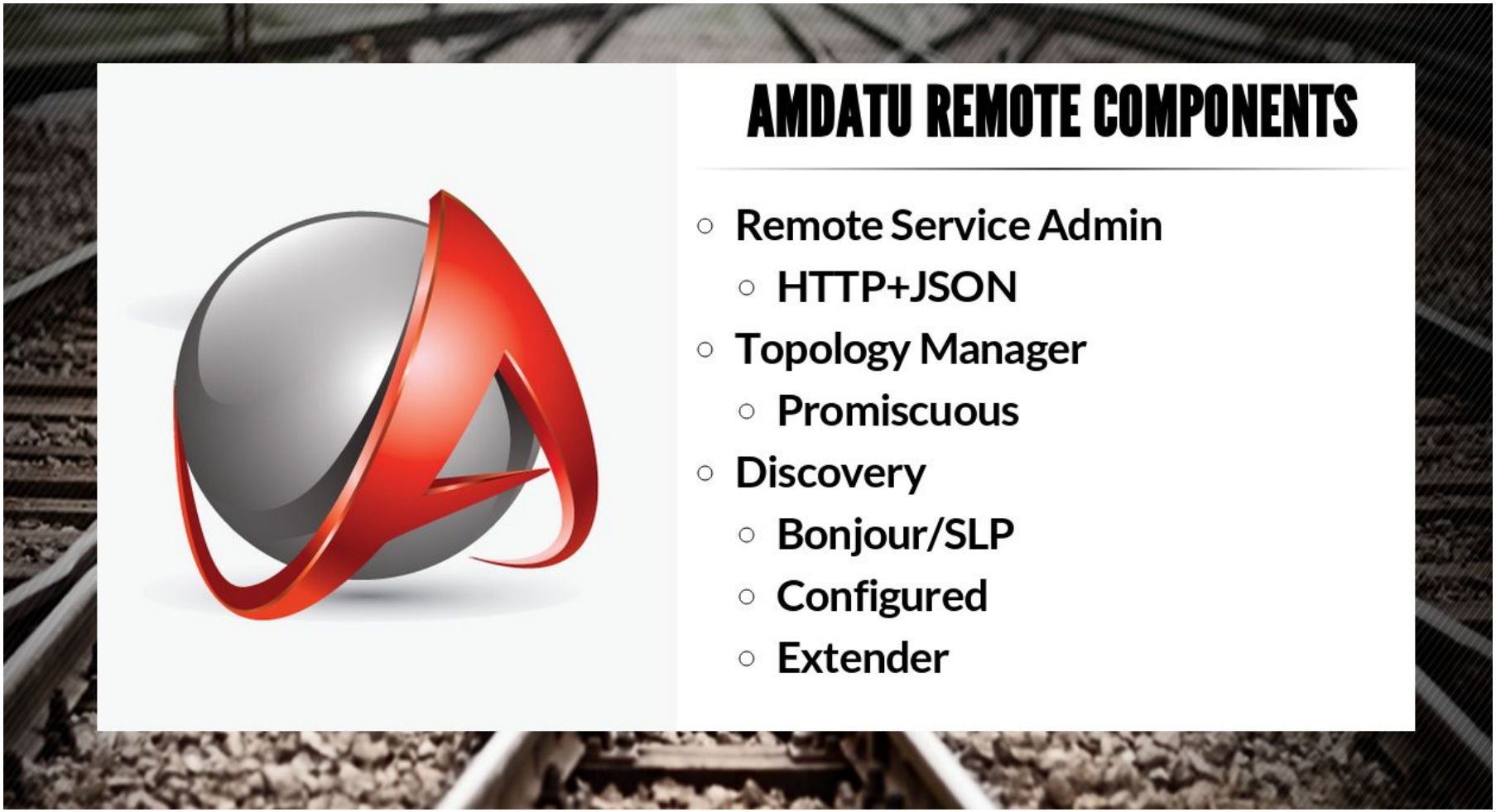
- ✓ Scale-out
- ✓ Transparent
- ✓ Pluggable
- ✓ Manageable
- ✓ Reusable


THE AMDATU PROJECT
OSGi/Cloud components
Web/JAXRS support
Multi-tenancy
Remote Services
MongoDB client
...more
Open Source (AL2)

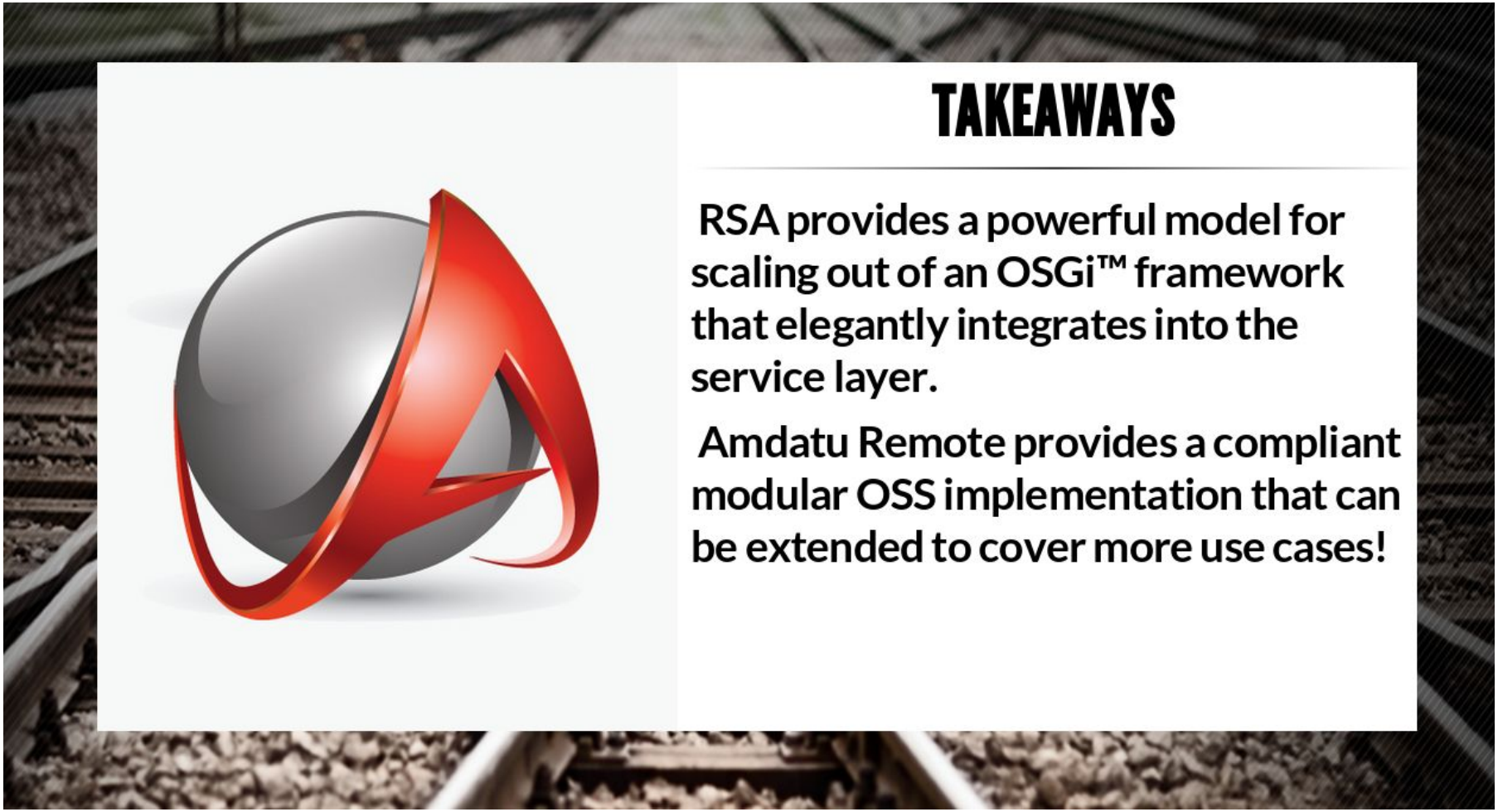


AMDATU REMOTE RELEASE

- RSA implementation
  - modular/simple
  - lightweight
- First release: R1
- RSA 1.0 compatible
- Supports RSA 1.1 (draft)
- RSA 1.1 RI candidate


AMDATU REMOTE COMPONENTS
Remote Service Admin
HTTP+JSON
Topology Manager
Promiscuous
Discovery
Bonjour/SLP
Configured
Extender

CODE & DEMO



TAKEAWAYS

RSA provides a powerful model for scaling out of an OSGi™ framework that elegantly integrates into the service layer.

Amdatu Remote provides a compliant modular OSS implementation that can be extended to cover more use cases!


THANK YOU!
More info:
http://www.amdatu.org
More talks:
The ultimate dependency manager shoot-out
Developing modular cloud applications with OSGi™