Trevor Dolby

IBM

IIBvNext and App Connect Enterprise 2: *Lightweight Integration for Microservice Architectures*

Important Disclaimers

IBM's statements regarding its plans, directions and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Important Disclaimers

- **IBM Confidential**. Unless specifically advised otherwise, you should assume that all the information in this presentation (whether given in writing or orally) is IBM Confidential and restrict access to this information in accordance with the confidentiality terms in place between your organization and IBM.
- **Content Authority**. The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- **Performance**. Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- **Customer Examples**. Any customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- **Availability**. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

Trademark acknowledgements

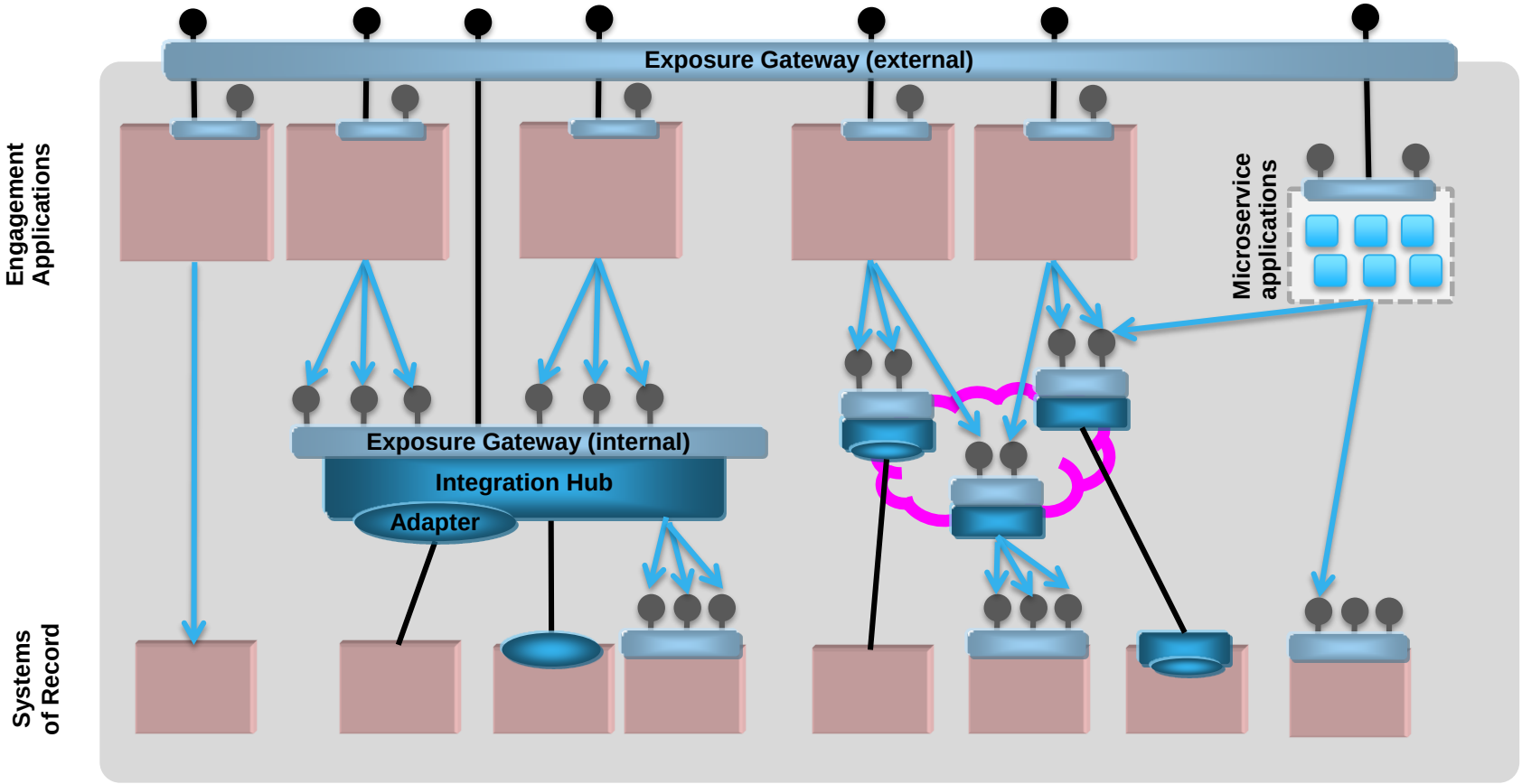
- IBM and the IBM logo are trademarks of International Business Machines Corporation, registered in many jurisdictions.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both
- Other company, product and service names may be trademarks, registered marks or service marks of their respective owners. A current list of IBM trademarks is available on the web at "Copyright and trademark information" ibm.com/legal/copytrade.shtml

Overview

- Introduce the concept of a Standalone Integration Server
  - What is a Standalone Integration Server (SIS)?
  - Describe the capabilities that these will offer
  - Engineering changes to support these

- Improvements to all Integration Servers as part of this work
  - Integration servers in an Integration Node
  - Integration/REST API and command architecture
  - Precompiled resources, persisted overrides etc
  - “Herding” integration server processes

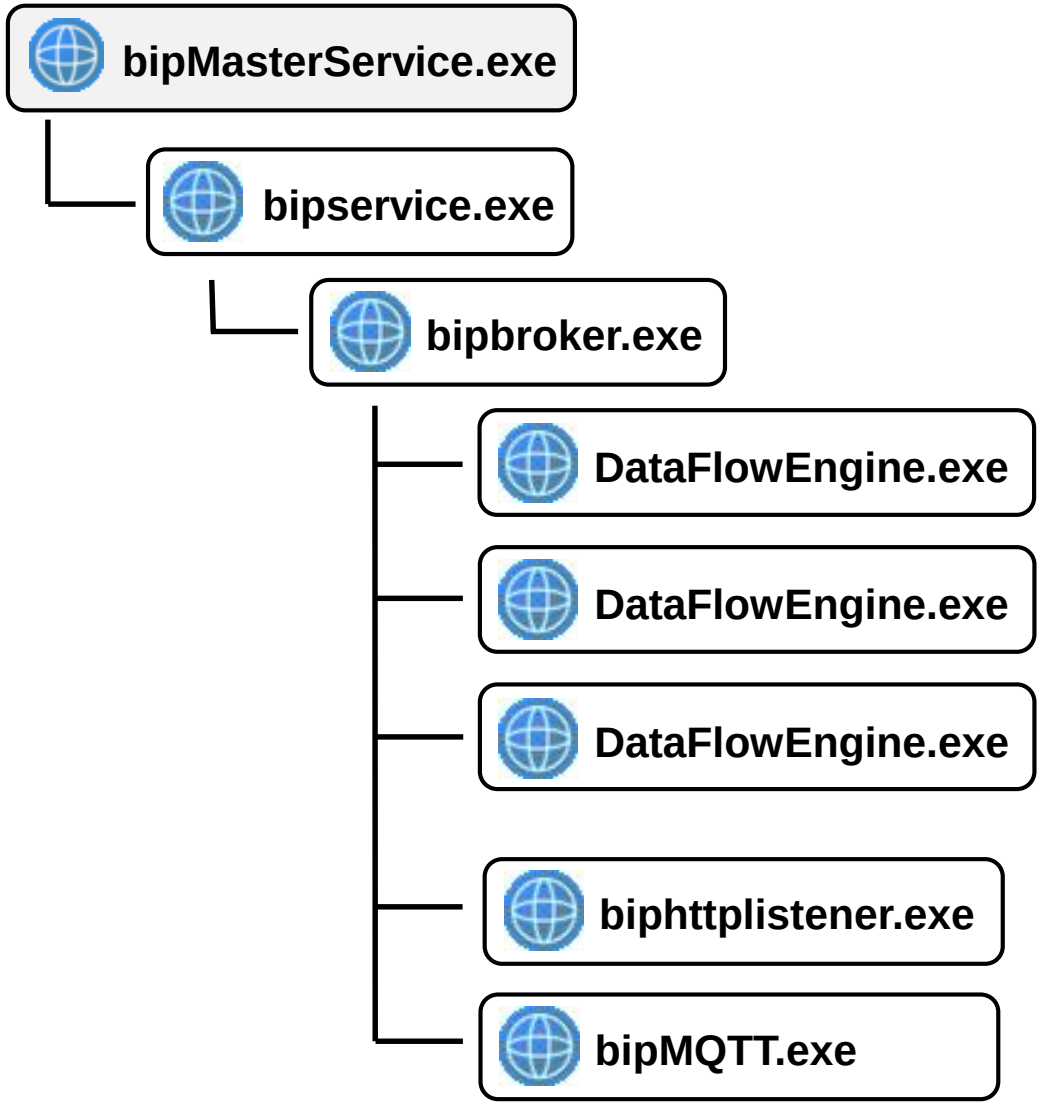
Evolving Integration architectures and the impact of Microservices



Reminder

IIB V10 Integration Node Structure

Process Overview

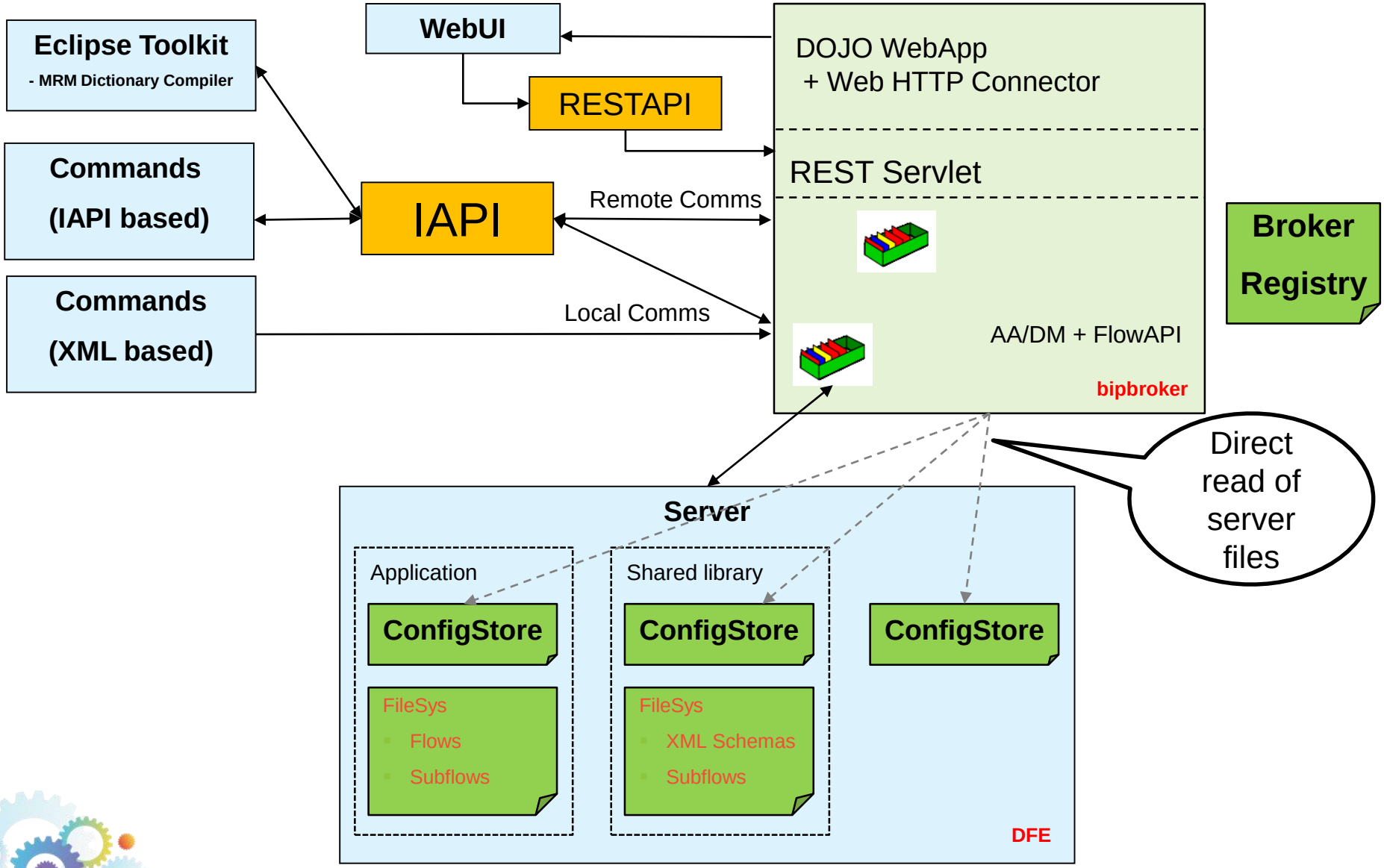


Process Overview Notes

- Process launch order:
  - When an Integration Node is started bipservice is launched
  - bipservice launches the bipbroker process
  - bipbroker launches the HTTP Listener
  - bipbroker launches its MQTT Server
  - bipbroker launches a DataFlowEngine (DFE) for each server
  - Any launching process monitors that process

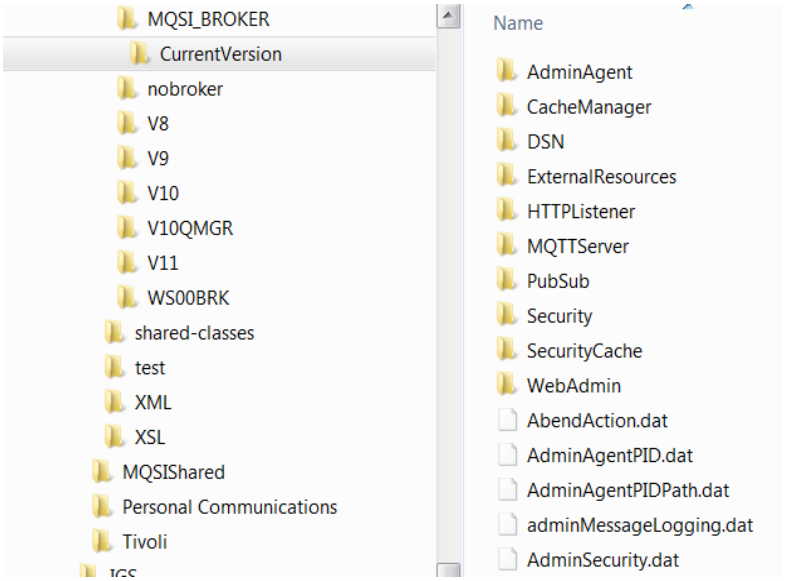
- Main process usage:
  - bipbroker handles all of the administration for a Node
  - DFE processes handle all message processing for an Integration Server
  - Integration servers cannot be administered without bipbroker.

IIB V10 Administration/Deploy chain overview

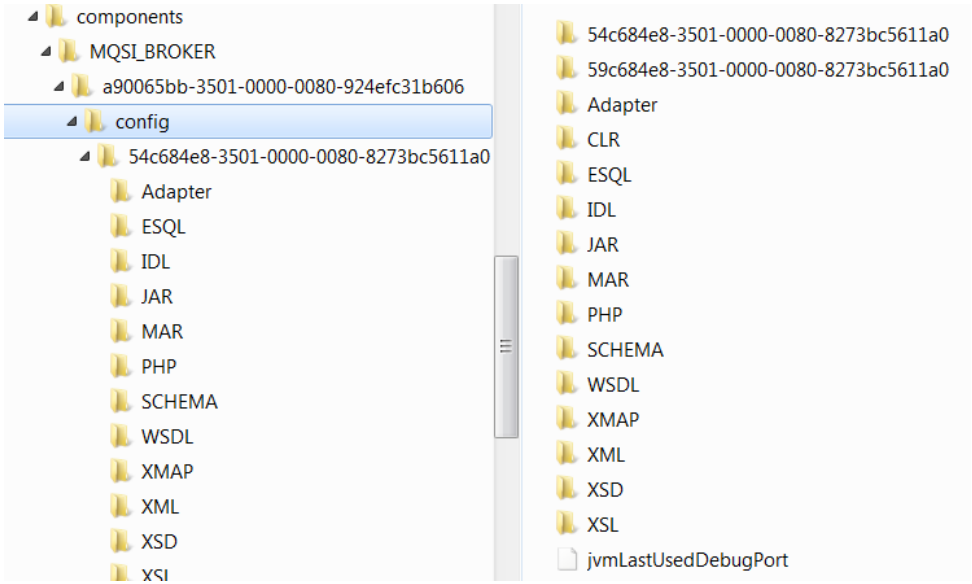


Configuration Storage

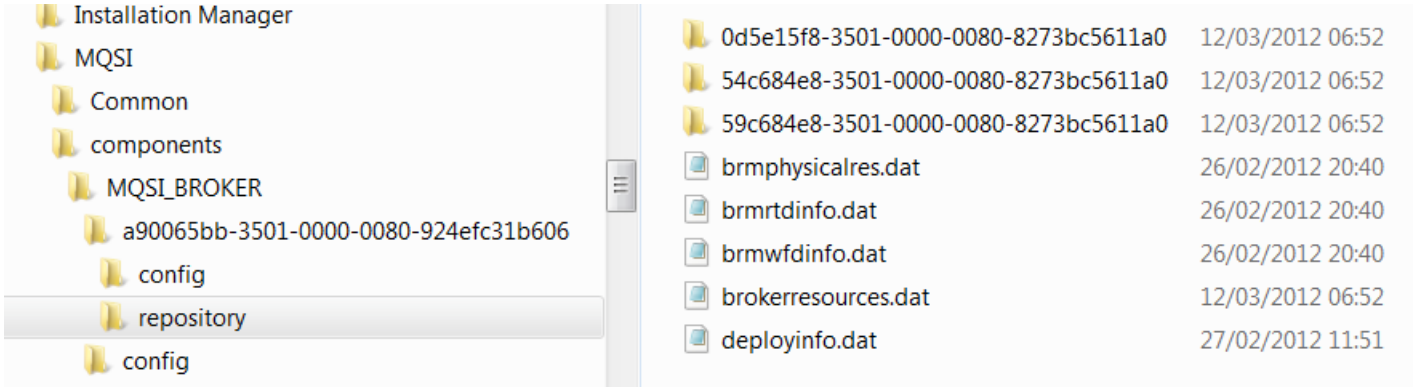
Registry



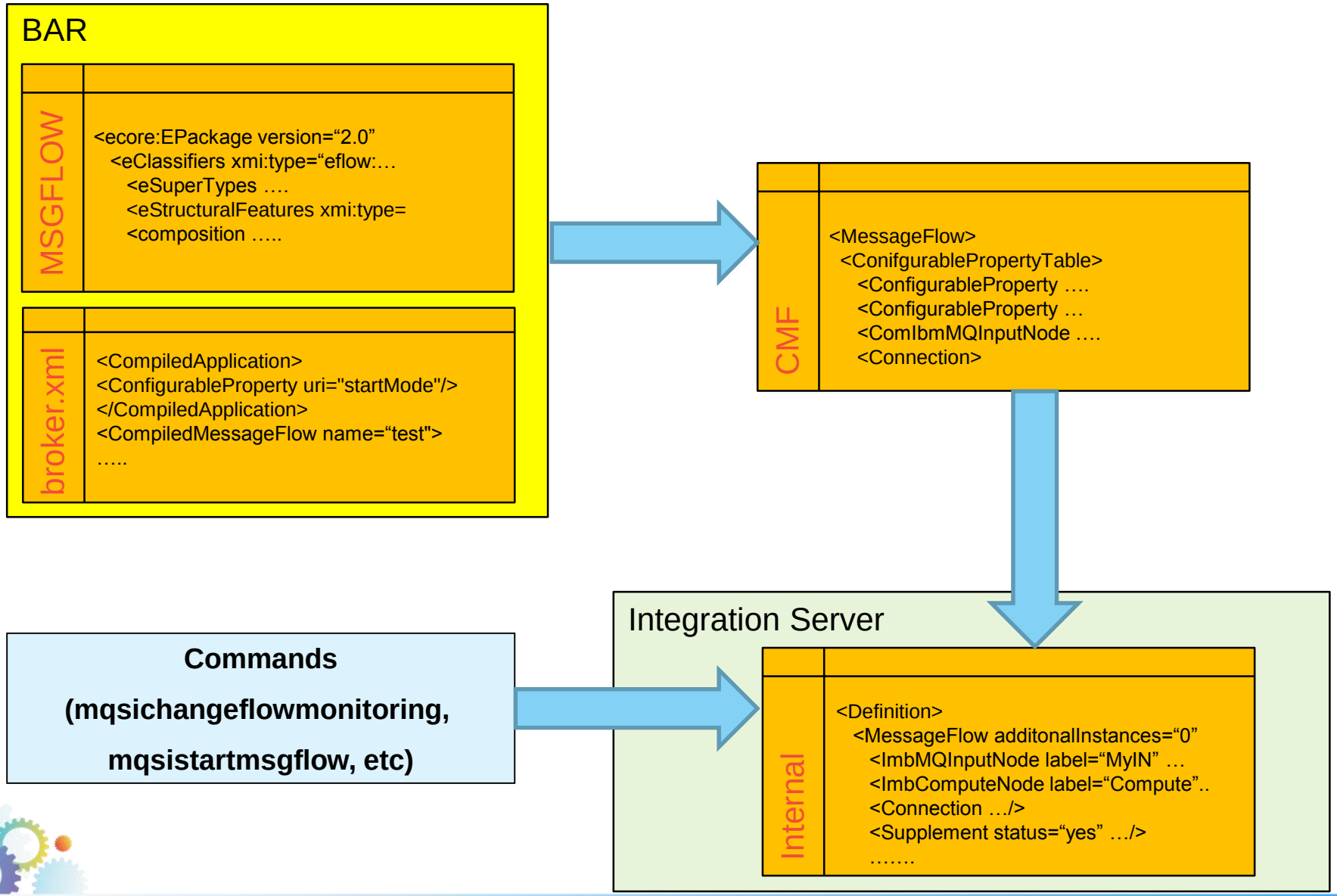
EG resources directory



Config Stores



Flow runs from translated/combined artefacts



Creation/Setup: Usually scripted

```
mqsicreatebroker TESTNODE
mqsicreateconfigurableservice TESTNODE -c EmailServer -o myEmailServer -n .... -v ...
mqsichangeproperties TESTNODE -b httplistener -o HTTPListener -n startListener -v false
mqsicreateexecutiongroup TESTNODE -e default
mqsichangeproperties TESTNODE -e default -o ComIbmJVMManager
   -n jvmMaxHeapSize -v 1048576000
mqsichangeproperties TESTNODE -e default -o ExecutionGroup
   -n httpNodesUseEmbeddedListener -v true
mqsichangeproperties TESTNODE -e default -o HTTPConnector
   -n explicitlySetPortNumber -v 8085
mqsistopmsgflow TESTNODE -e default
mqsistartmsgflow TESTNODE -e default
mqsideploy TESTNODE -e default -a solutionBAR1.bar -w 500
mqsideploy TESTNODE -e default -a solutionBAR2.bar -w 180
mqsideploy TESTNODE -e default -a solutionBAR3.bar -w 250
```

Common observations

- Extremely capable, industrial-strength infrastructure. But:
  - Overhead is more costly than desired.
  - Set-up is more complex than desired.
  - Development/Test environments quite large
  - Current process centred around production environments

- Container/Cloud-based systems have different requirements
  - Ideally would run just the DataFlowEngine

Standalone Integration Servers

Standalone Integration Servers (SIS)

- Run direct from the command line
  - Similar to StrongLoop, Mongo, etc
  - No create step.
  - No deploy step necessary
  - Process is not owned by any other process (or Integration Node)
    - Can be managed by container-based admin (Kubernetes, etc)

- Basic parameters for frequently used settings:
  - Can be given a name under which it runs (should be unique)
  - Default MQ queue manager name
  - http port for HTTP nodes
  - Named event log file rather than logging to syslog/Event Viewer
  - JVM options so that min/max heap size can be specified etc

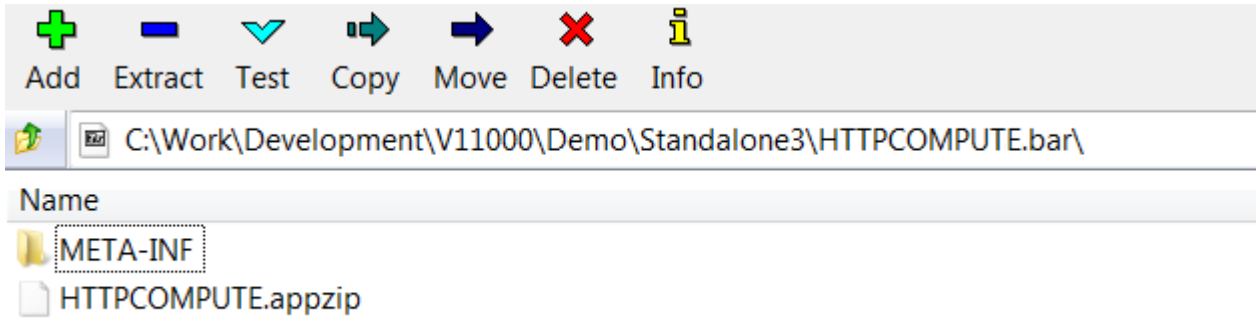
Standalone options

IntegrationServer
--work-directory C:\work\myServer
--event-log C:\work\myServer\syslog.txt
--mq-queue-manager-name MQV9_QM
--http-port-number 7800
--dev-ui-port 4800
--java-{jvm option name) jvmOptionValue
--service-trace
serverLabel

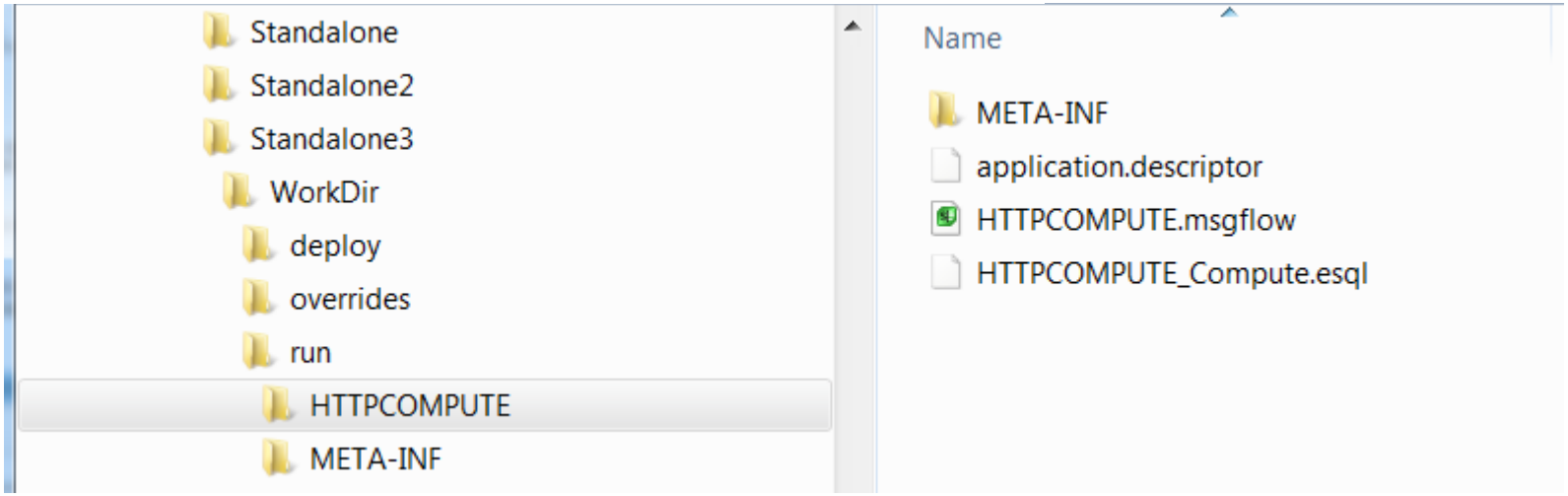
Current options for the Beta

Unzip and Go

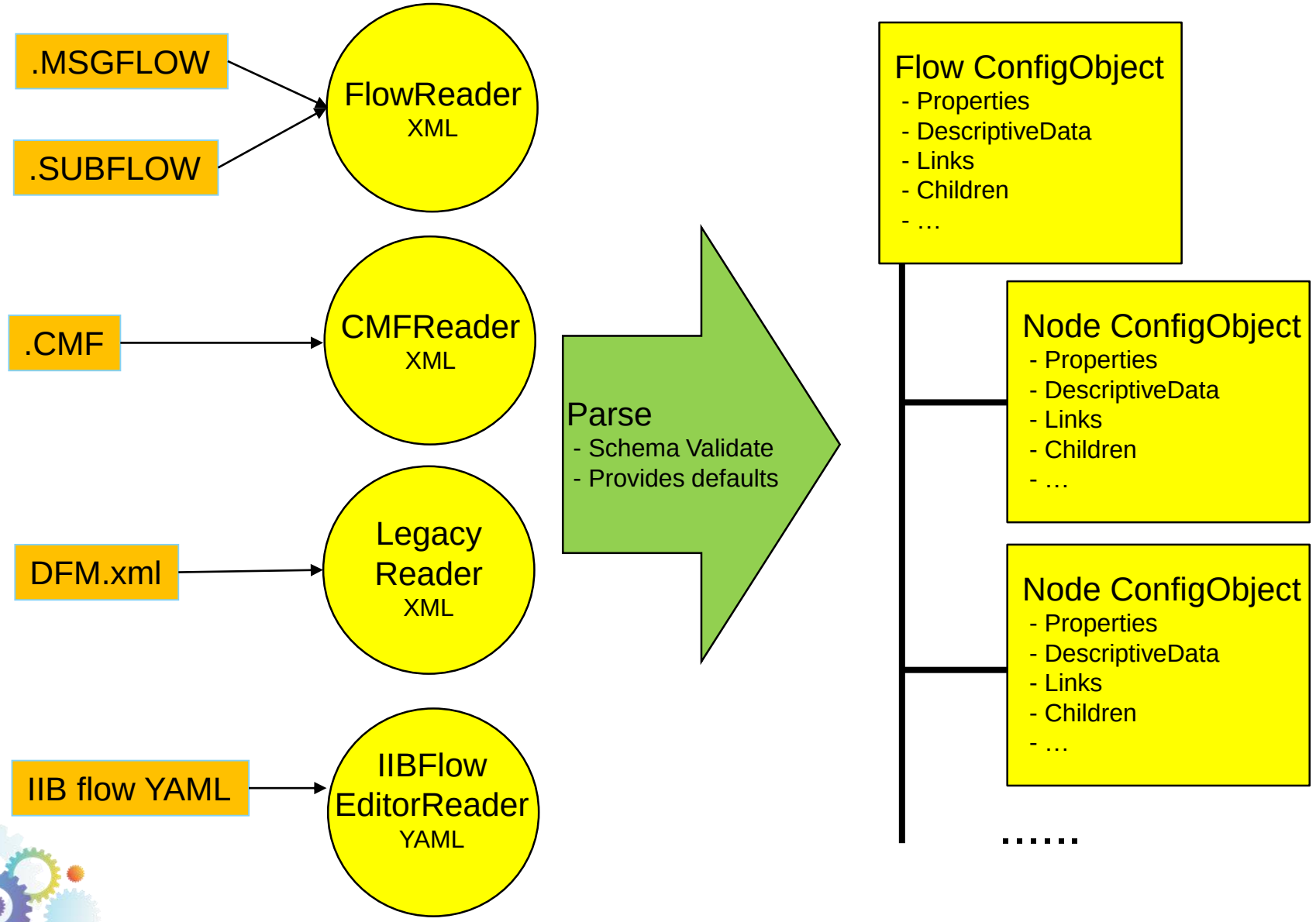
- "run" directory reflects the structure of the BAR files



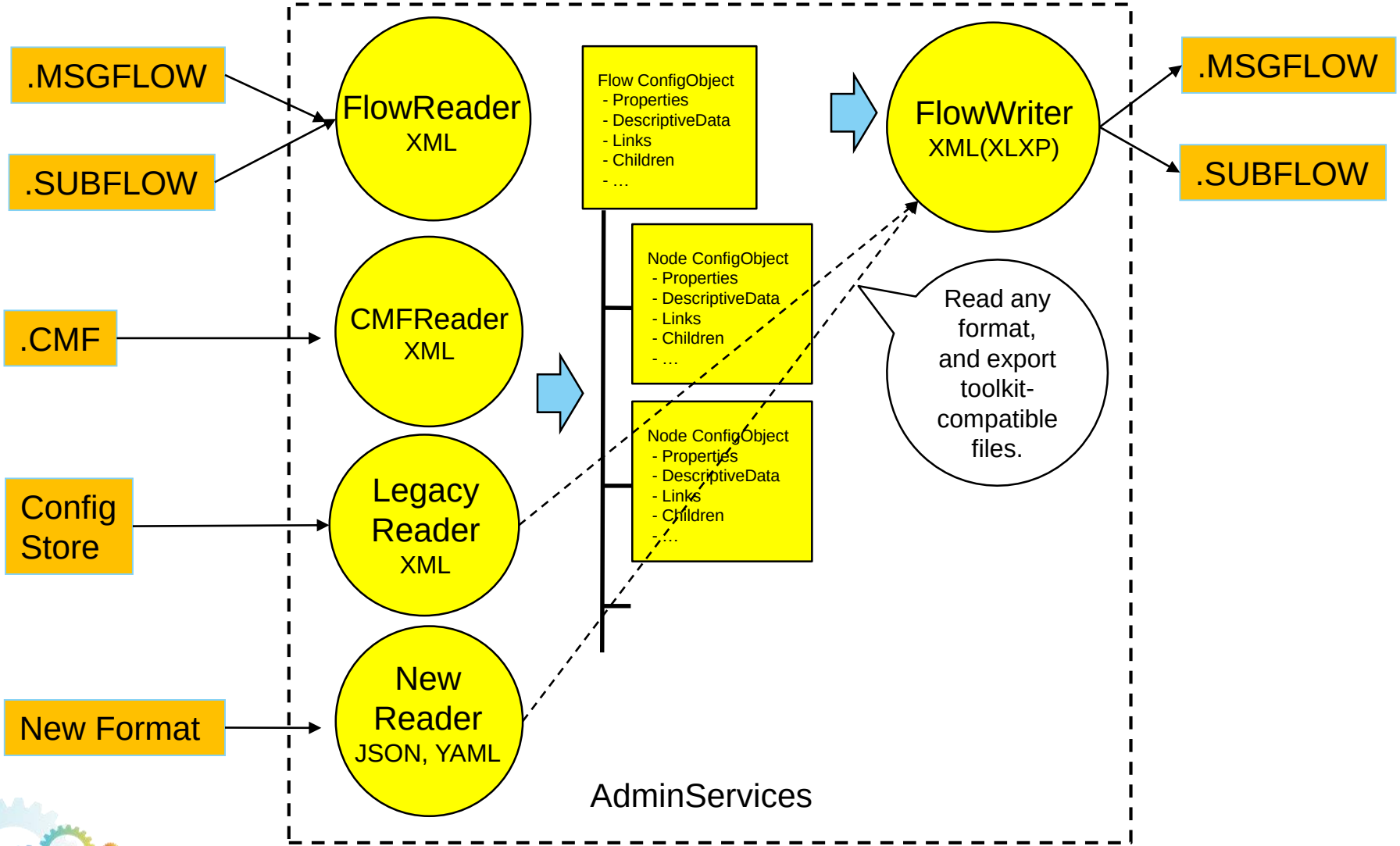
No internal directory structure



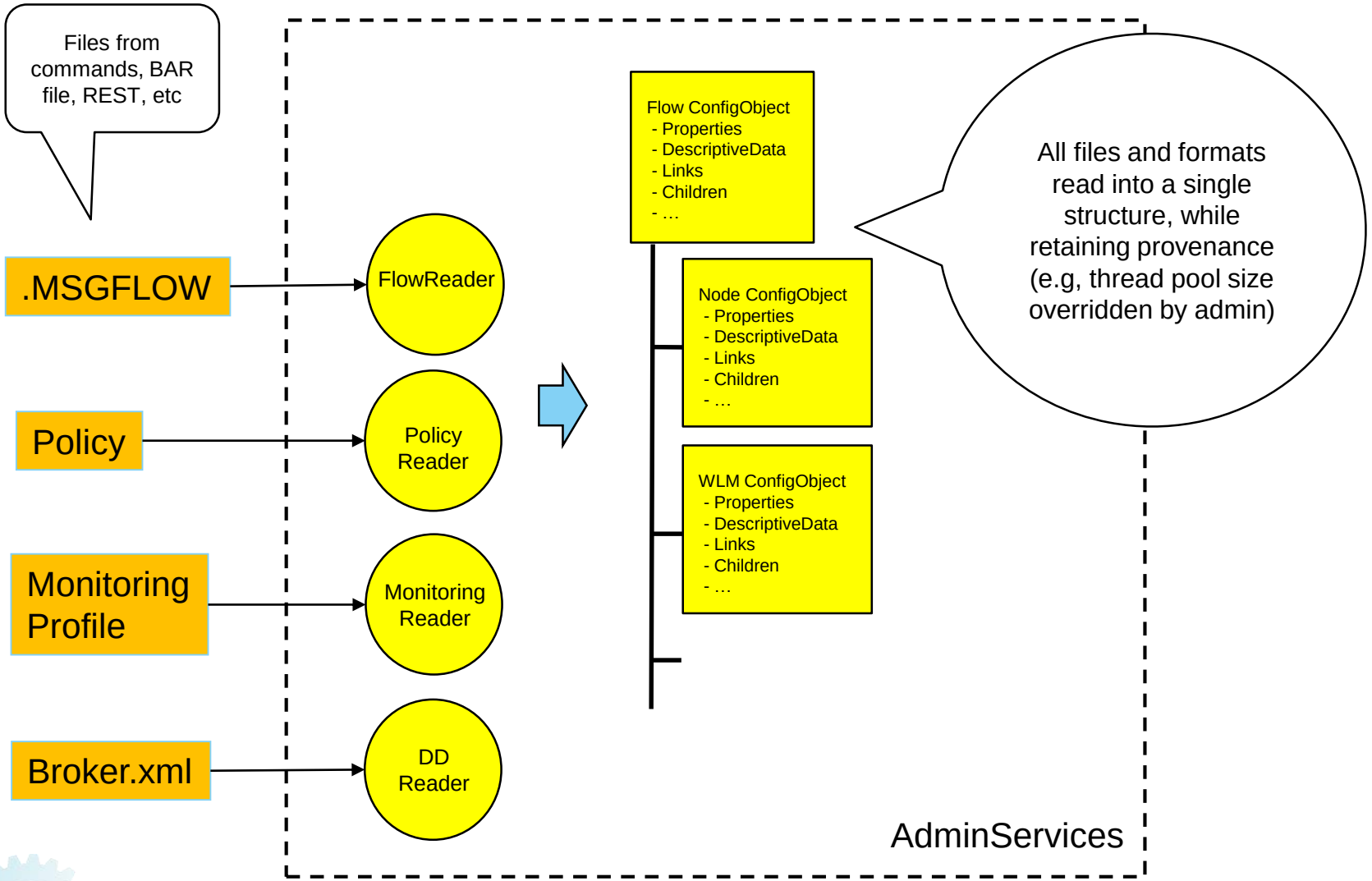
Flow/Subflow readers



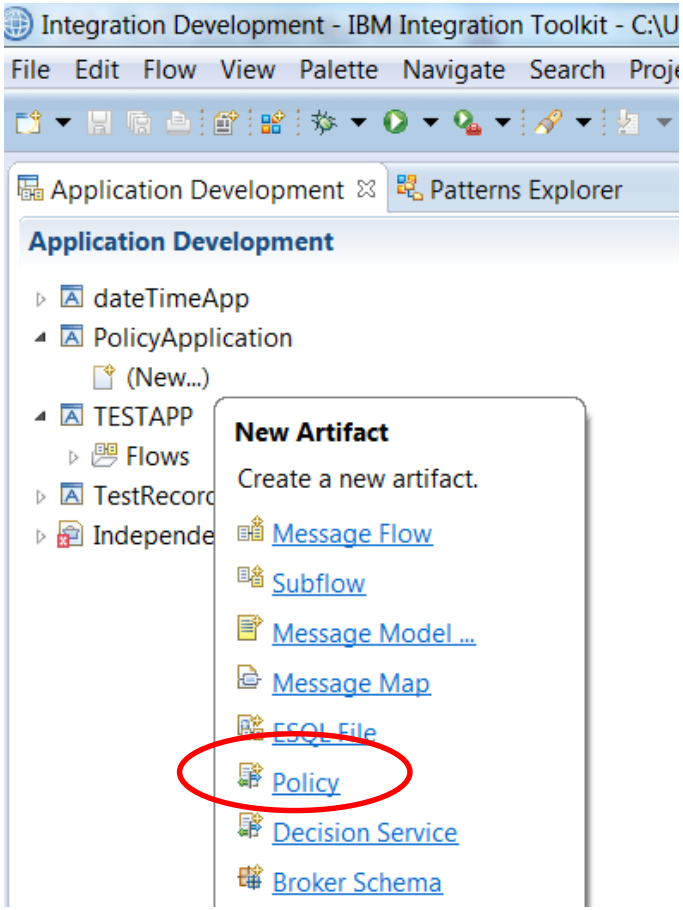
Any file format to any other

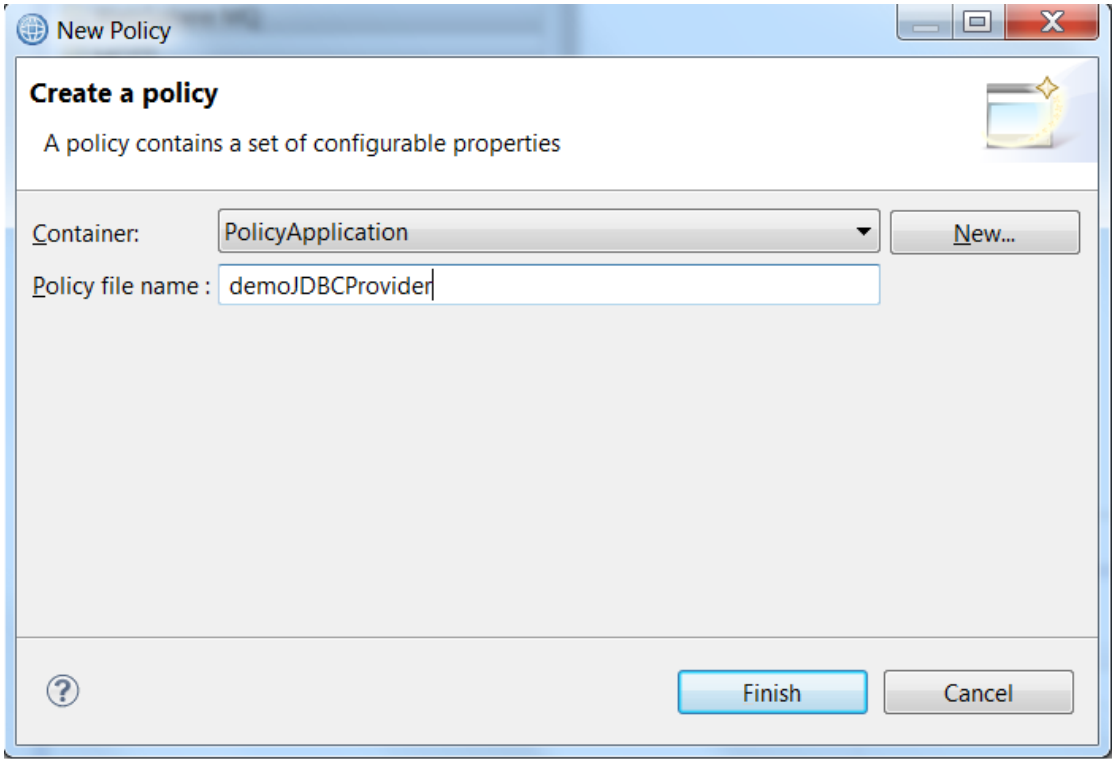


Overrides and deploy descriptor

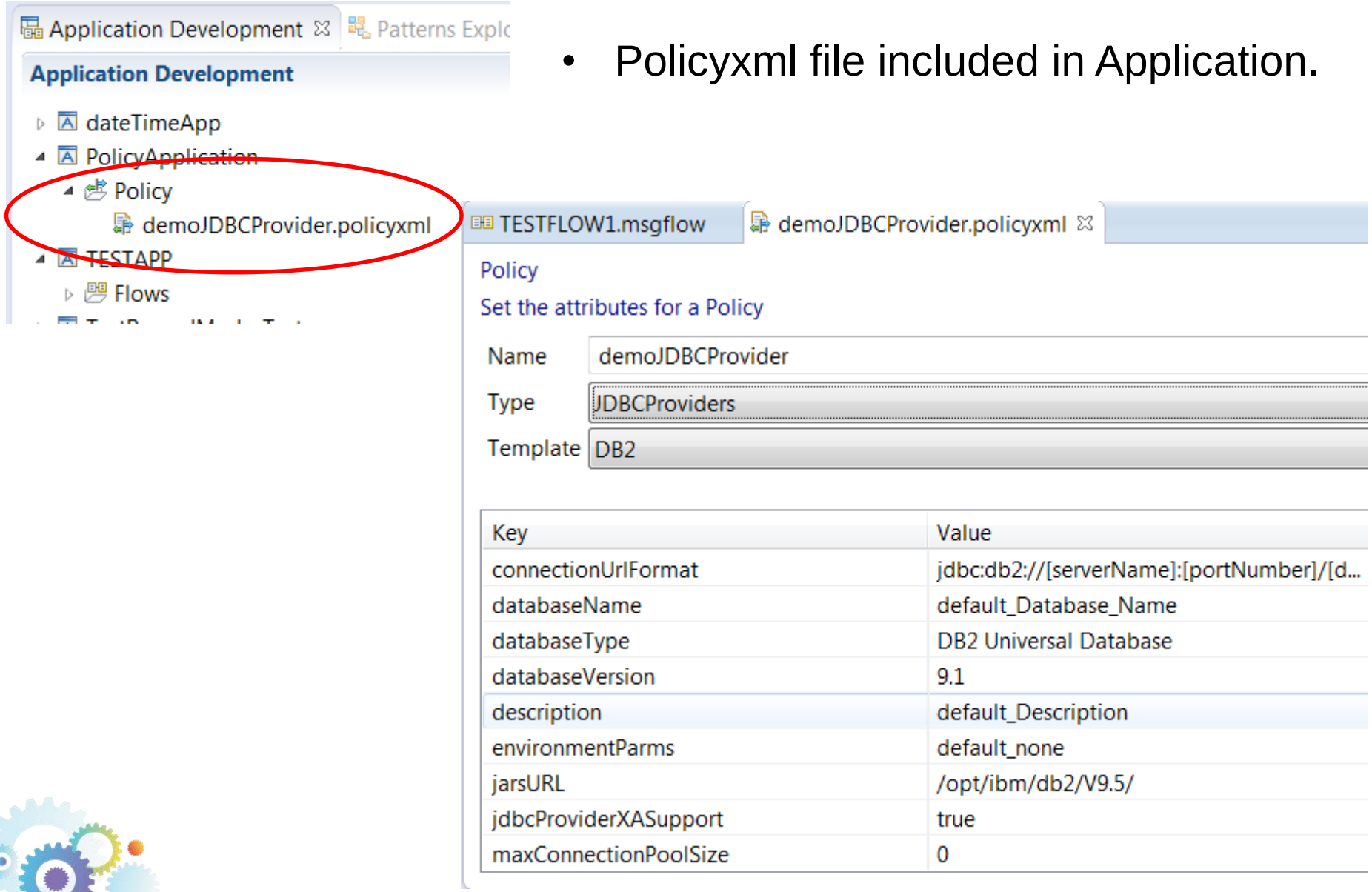


Policy creation: Toolkit (1)



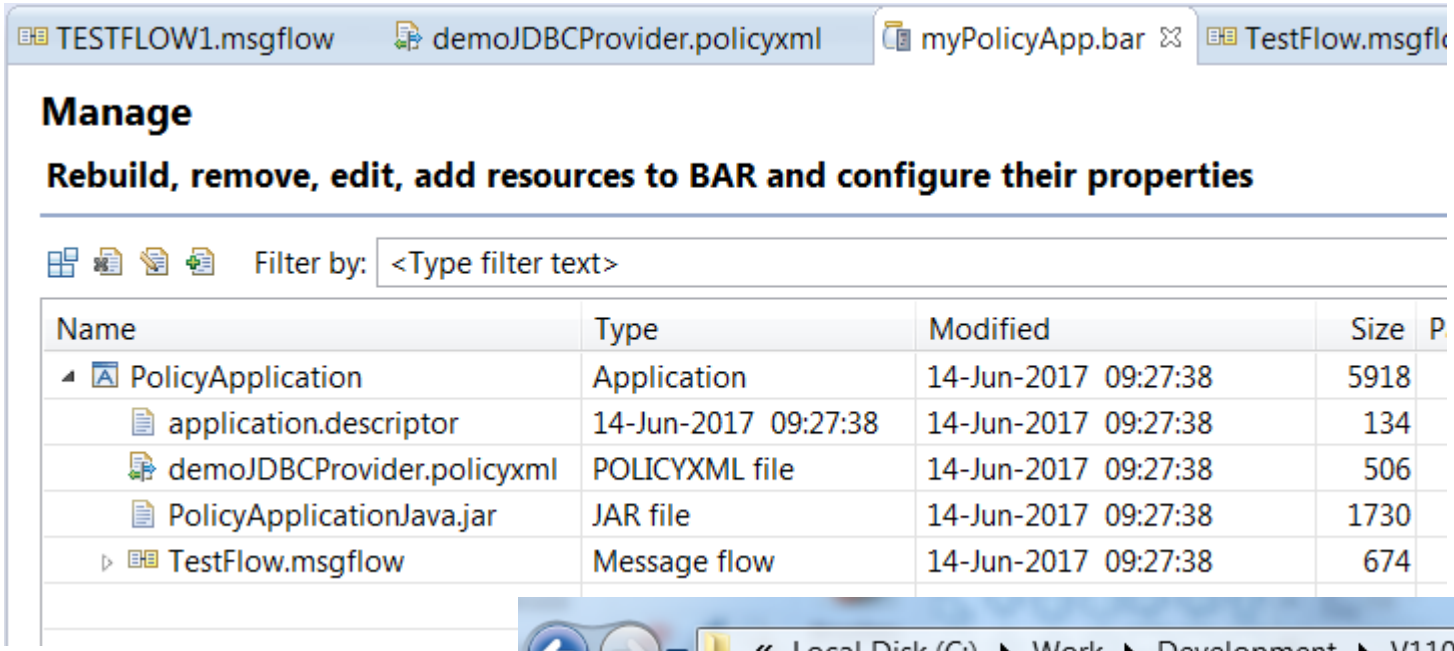


Policy creation: Toolkit (2)

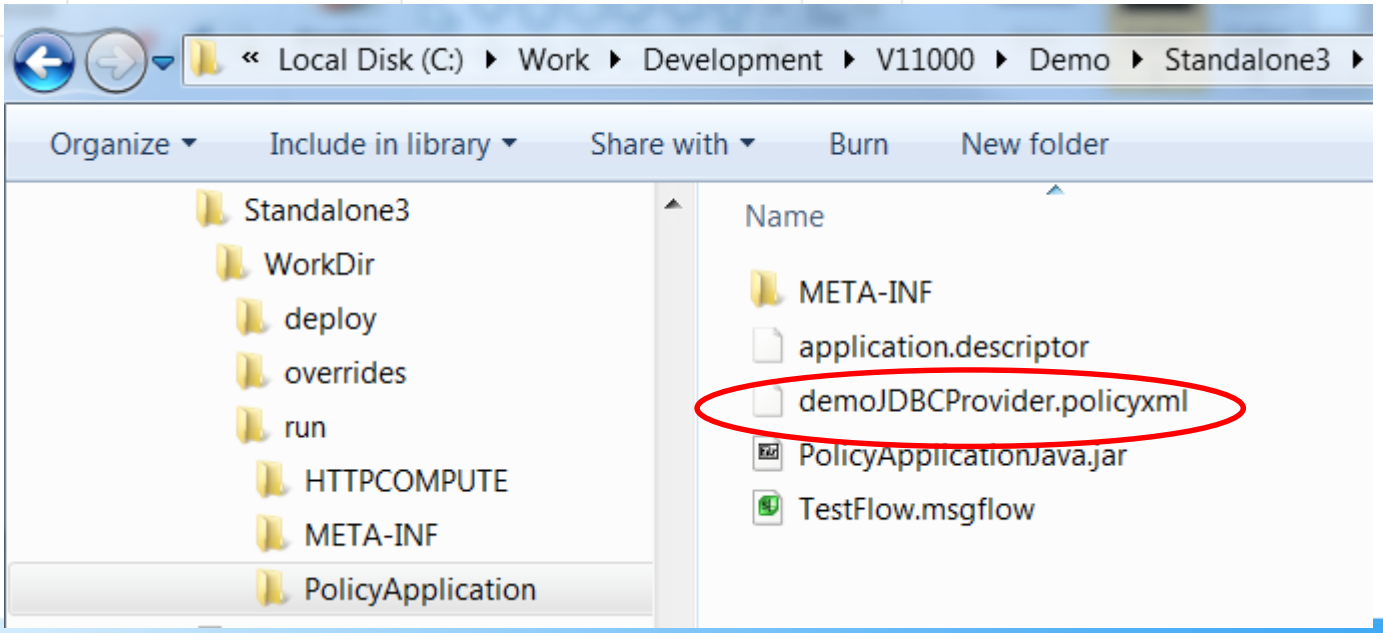


- Policyxml file included in Application.

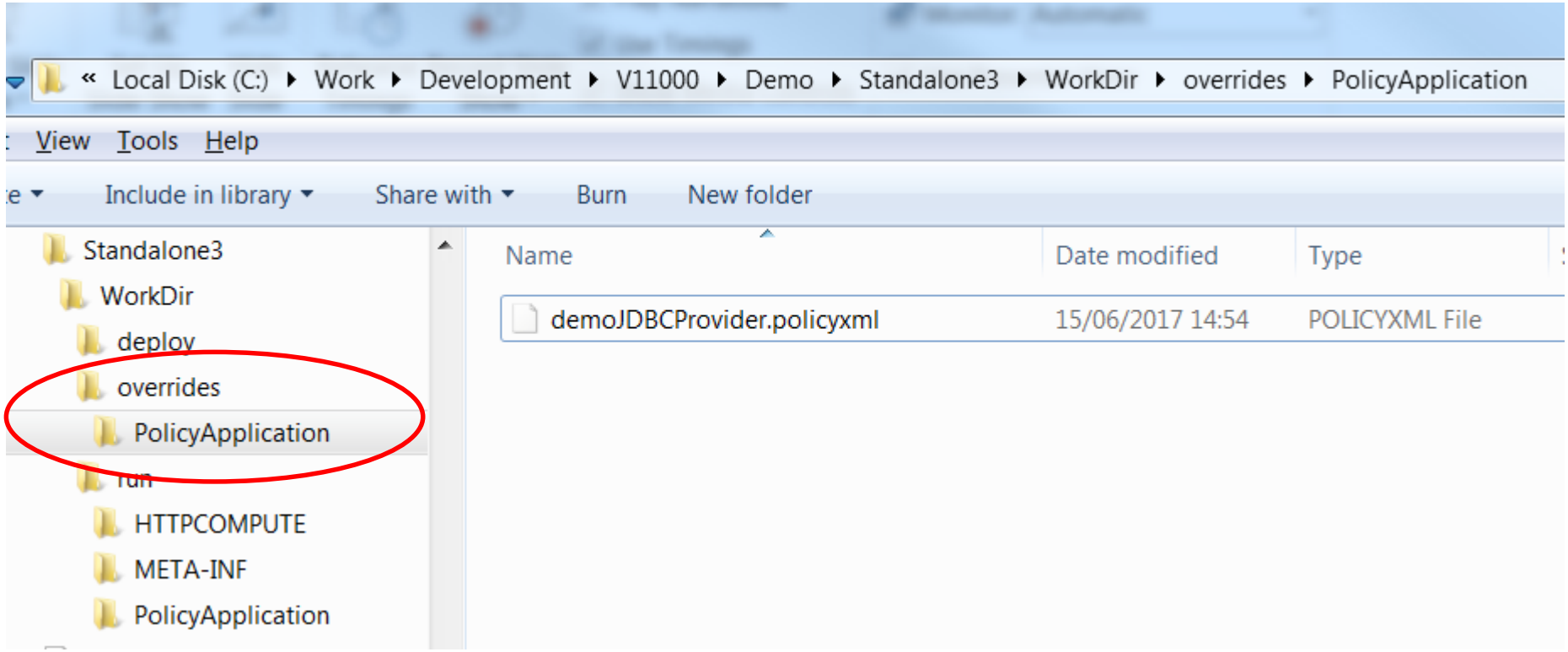
Policy creation: “Unzip and Go”



- SIS can process policyxml files in startup/ overrides directory



Overridden Policy



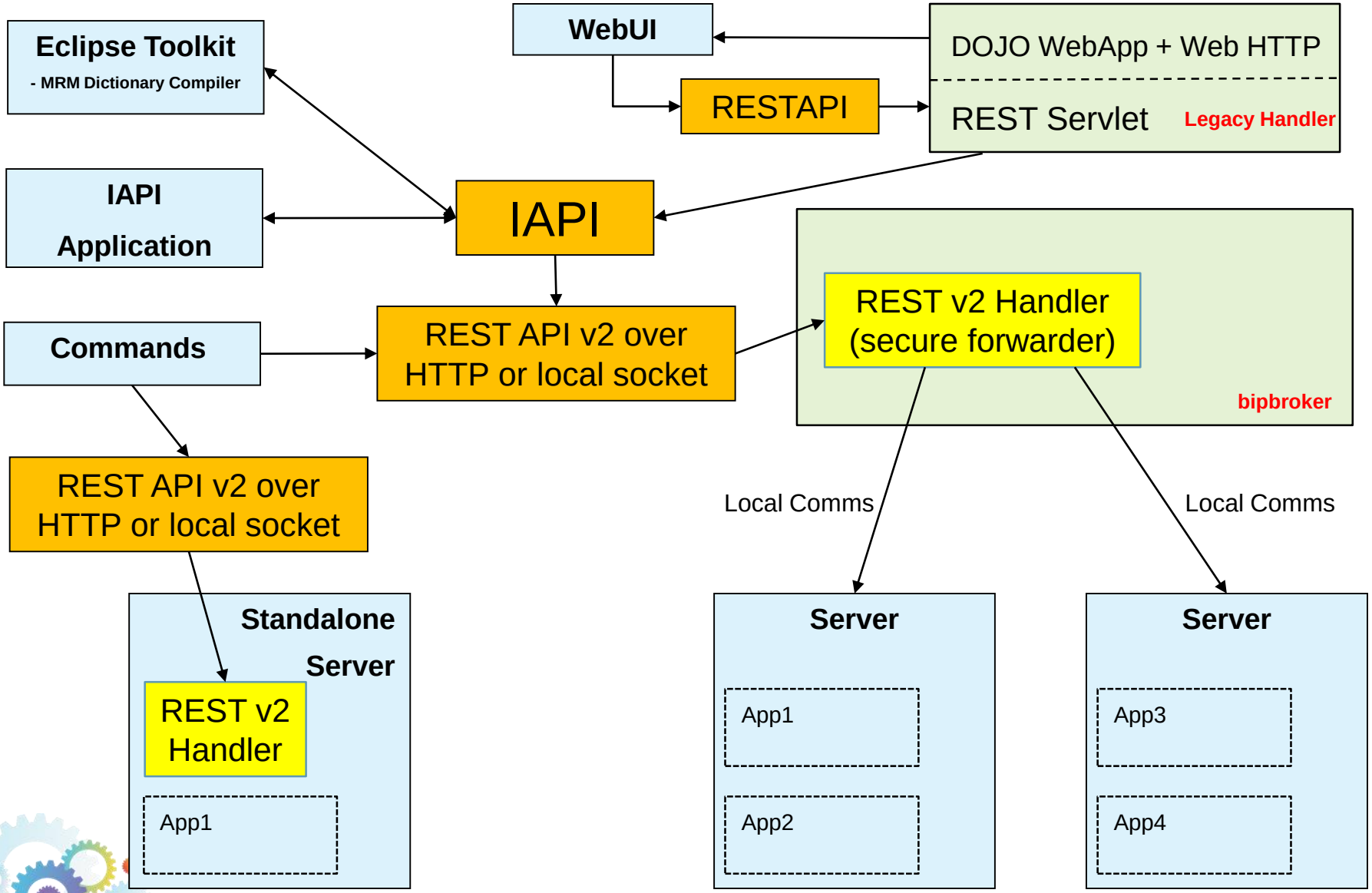
- Standalone servers start with an overrides directory
  - Has the same hierarchy as the source directory.
- Overrides take precedence over the source material
- Updated policy can be placed in the application directory

Engineering

- Common approach for all configuration.
  - Bars, Flows, subflows, policy, overrides etc
- Specific readers parse input documents
  - Do not have to be tied to a specific format
  - Create configuration objects using schema defaults
  - Common validation approach irrespective of format.
- Configuration model provides:
  - Common format for all configuration
  - Common “diffing” approach to determine what has changed
  - Transformation services from one format to another
  - Run-time objects can be sourced from multiple documents

Integration Nodes and servers

Integration Nodes in vNext



Integration Servers

- Internal design shared by all Integration servers
  - Standalone server not a separate codebase
  - Each Integration Server will have a source directory
    - Deployed artefacts will “land” in this directory in the unzipped form

- bipbroker will revert back to being just a:
  - Router for administration requests
    - Communication can still be via local comms rather than over REST
  - Monitor/”Herder” for its Integration servers
  - Manage node level configuration
  - Continue to provide HA functionality.
  - Continue to provide administration security

Updates/Redeploy

- Servers can run off source in an “unzip and go” scenario
  - Source retrieved from a repository
  - Unzipped to a target directory
  - Standalone Integration server is started on that directory

- Servers will support the concept of an update:
  - In the beta, standalone servers use a deploy directory.
  - Going forward this will be a REST request to bipbroker (if using an Integration Node) or the server itself.
    - POST, PUT, PATCH
  - BAR files that add/update existing resources
  - Property level changes similar to existing commands

Differential deploy

- In IIBv10 a redeploy can have a large impact on resources:
  - Applications/Library have all existing contents deleted.
  - Post-deploy overrides are lost.
  - Maps and Schemas are recompiled
  - Flows interrupted for longer than desired.

- Re-engineered server improves this area:
  - Preservation of source structure allows targeted updates
  - Configuration model allows exact changes to be calculated.
  - Schema model allows scope of change to be described

Differential deploy possibilities

- Individual property changes can be made
  - "hot swapped in" if the property allows this.
  - Brief update under lock if needed.
- Individual message flow nodes can be changed
  - Node "re-starts" with a new configuration
- Flow level changes
  - Only the individual flow that is affected will be rebuilt.
- Application/Library level changes
  - Only affected resources are updated and changed
  - No "tear down" of all existing contents
- Post-deploy overrides are not lost
  - Eg, monitoring, statistics, policy attachments etc

Pre-compiled resources

- Maps and schemas are compiled on deploy in IIBv10
  - Time-consuming for large numbers of maps/schemas
- Compilation can now take place independently…

```
mqsipackagebar
  -a SIS_App_Shlib_WithPreComp.bar
  -w C:\devel\SIS_Scratch -k MapAndSchemaInShlib_App
  -y MapAndSchema_Shlib -i
```

Beta: -i used for compilation

Command output – Schema compilation

Generating runtime objects for '[MapAndSchemaInShlib_App]' in 'C:\devel\SIS_Scratch' ...

Shared Lib MapAndSchema_Shlib

Shared Library MapAndSchema_Shlib contains maps: [C:\devel\SIS_Scratch\MapAndSchema_Shlib\MapJson2CsvInShlib.map]

Shared Library MapAndSchema_Shlib contains schemas: [C:\devel\SIS_Scratch\MapAndSchema_Shlib\csvWithHdr.xsd, C:\devel\SIS_Scratch\MapAndSchema_Shlib\IBMdefined\CommaSeparatedFormat.xsd, C:\devel\SIS_Scratch\MapAndSchema_Shlib\shlibXml.xsd]

App MapAndSchemaInShlib_App

App MapAndSchemaInShlib_App references other library MapAndSchema_Shlib

Have 1 schema files to compile

Wrote XLXP BIR file: C:\devel\SIS_Scratch\MapAndSchemaInShlib_App\$mqsiApplication.bir

Information: No .xsd with DFDL annotations found in input, will not create a .dpif

Have 3 schema files to compile

Wrote XLXP BIR file: C:\devel\SIS_Scratch\MapAndSchema_Shlib\$mqsiLibrary.bir

Wrote DFDL PIF file: C:\devel\SIS_Scratch\MapAndSchema_Shlib\$mqsiLibrary.dpif

Command output – Map compilation

Generating map: C:\devel\SIS_Scratch\MapAndSchemaInShlib_App\AppMapShlibDfdl2xml.map

Generating map: C:\devel\SIS_Scratch\MapAndSchema_Shlib\MapJson2CsvInShlib.map

Generated, now packaging BAR

Packaging content from 'C:\devel\SIS_Scratch ...

Successfully added file '$mqsiApplication.bir' to the BAR file.

…..

Successfully added file 'xmap_default__AppMapShlibDfdl2xml.ser' to the BAR file.

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266.class' to the BAR file.

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common$ConvertToXCIState.class' to the

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common$InitParam.class' to the BAR file.

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common$KeyTable.class' to the BAR file.

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common$NodeKind.class' to the BAR file.

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common$subText.class' to the BAR file.

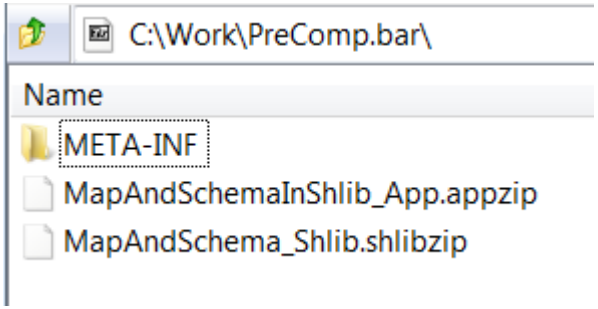
Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common$TunnelParam.class' to the BAR file.

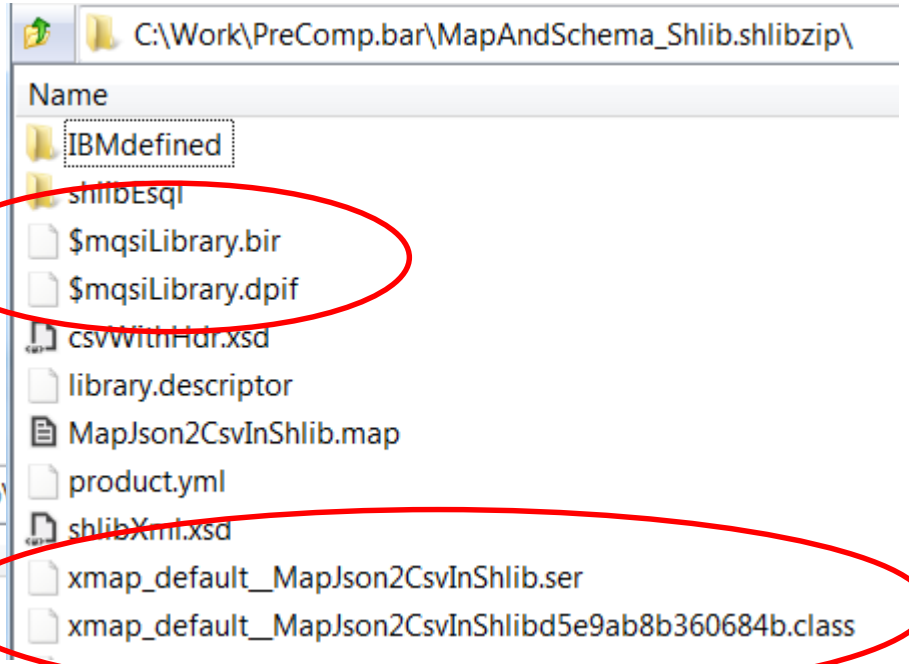
Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common.class' to the BAR file.

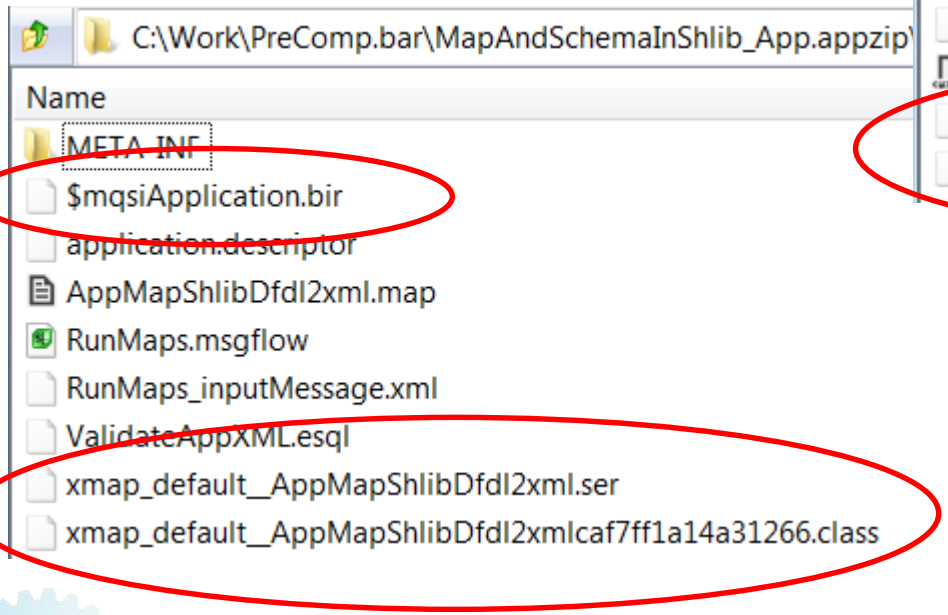
Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common_Partition0.class' to the BAR file.

Successfully added file 'xmap_default__AppMapShlibDfdl2xmlcaf7ff1a14a31266_Common_Partition1.class' to the BAR file.

Command output: The BAR







- Compiled resources unzipped
- No compilation on start-up
- Server runs off them
- Included in differential deploy

Future directions post-Beta

What's coming up?

- One "work-path" directory for a standalone integration server.
  - Source and overrides sub-directories (no deploy)
  - Log directory for traces and other diagnostics.
- Advanced configuration:
  - Configuration/policy files in the start-up/overrides directory
  - Fully replace configurable services.
- REST API ("/apiv2")
  - Will be able to present "live" information from the process
  - Current /apiv1 is limited to configuration information only
- Separable map/schema compilation
  - Enable in toolkit and other externals

Other possibilities

- Formal separation of deploy/start
  - Resources can be “deployed” without trying to start them
    - Deploy will succeed even if dependencies are missing
  - Start with validate the dependency chain
    - If valid the resources will be started

- Impact analysis
  - Report resources that are linked to each other
  - Report artefacts that are currently in use
  - “Pre-deploy” step to show what will be impacted by changes

And we haven't forgotten …

- Standalone servers and:
  - Global cache
  - Administration security
  - Modes of operation
  - HA scenarios
  - Existing administration WebUI
- Existing Integration/REST API and command usage
- Migration from previous versions
  - Support for server-level resources
  - MessageSets, Adapters in Shared Libraries

Thank you and Questions?