Introducing the Liberty collective: A cloud-enabled application server management foundation

Chris Vignola

Evolutionary forces driving the IT industry toward cloud computing increase the pressure on application server management solutions to both accommodate traditional infrastructure deployments and enable cloud-based deployments. Lightweight, standards-based, flexible scalability are among the necessary hallmarks of an architecture that embraces this duality. Liberty collectives deliver next generation application server management for the IBM® WebSphere® Application Server Liberty profile both inside and outside the cloud.

Introduction

Cloud computing is on the rise. The ever-increasing connectedness of the world is driving up demand for online services. The IT industry in turn is responding to this demand with new delivery models centered around the cloud computing concept, which promises to dispense these services at the right time, at an acceptable cost, and on the right infrastructure. These models exploit techniques such as virtualization, dynamic provisioning, and elastic scaling to achieve these goals. The result is a large and growing number of application servers in a flexible and dynamically changing environment — the cloud environment.

The ultimate goal of the cloud environment is to be automatic and self-managing. Achieving this goal requires that the software providing the foundation for the cloud environment have a way to manage the lifecycle and operational concerns of large numbers of heterogeneous application servers and other supporting elements. While cloud foundation software might typically provide a generic framework to manage application servers, the framework must necessarily be customized to leverage the indigenous management interfaces of specific application server types.

Not only must the cloud foundation software make use of indigenous management interfaces, so too must IT staff, who have a requirement to manage application servers outside of — or alongside — the cloud environment. This need arises out of numerous possibilities, including filling gaps not presently addressed by cloud foundation software, intervening to work around cloud foundation failure, or managing application servers outside of the cloud entirely. An effective indigenous
systems management foundation would provide consistency across all use cases. For the IBM
WebSphere Application Server Liberty profile, this indigenous foundation is the Liberty collective.

The Liberty collective is the latest innovation for WebSphere Application Server systems
management, available in version 8.5.5. It is a systems management architecture designed from
the ground up exclusively for the Liberty profile and the cloud. The Liberty collective provides an
effective indigenous systems management foundation engineered for consistency across both
cloud and non-cloud deployments.

The Liberty collective provides the best practice deployment pattern for new multi-server Liberty
deployments. Among the many advantages it offers, the Liberty collective provides:

• Lightweight cluster support for the Liberty profile.
• Control point for automated deployment of Liberty archive packages to multiple host systems.
• Secure, scalable, highly available operational point of control for all Liberty servers in the
  same administrative domain.
• A simple control point for monitoring operational status of all Liberty servers in the same
  administrative domain.
• An easy way to determine inventory of all Liberty servers in a given administrative domain.
• Secure connections to all Liberty servers in the same administrative domain without having to
  manage passwords on any server, except the control points (collective controllers).

Looking ahead, you can expect the Liberty collective to be the foundation for other multi-server
functions and an integral part of cloud-based deployments.

Anatomy of a Liberty collective

A Liberty collective is an administrative domain for a group of Liberty profiles (servers). A Liberty
server in a collective is called a collective member. A Liberty collective has one or more
members; typically many. A Liberty collective has a control point through which administrative
clients are able to discover and securely operate on any constituent collective member. This
control point is called a collective controller. The Liberty collective controller can be configured
for both scalability and high availability of the administrative domain.

Here are the elements that make up a Liberty collective:

• **Collective controller** is a control point for a Liberty collective. It is a Liberty server
  configured with the Liberty collectiveController-1.0 feature. The controller holds information
  about collective members. A collective controller exposes MBeans for administrative clients
  to interact with collective members through a REST-based interface. A collective can be
  configured with one or more collective controllers, depending on the scalability and high
  availability requirements of the administrative domain. A collective controller is also a
  collective member; it can be a member of one and only one collective.
• **Collective member** is a Liberty server in a Liberty collective. A collective member is
  configured with the Liberty collectiveMember-1.0 feature. A collective member shares
  information about itself with the collective controllers in the same collective. This information
includes network location, security information, and operational status. The collective controllers use this information to carry out and delegate operations to collective members. A collective member can be a member of one and only one collective.

- **Collective host** is a host system (operating system instance) on which one or more collective members reside. A collective host can have any number of collective members, each belonging to the same or different collectives.

- **Admin clients** are any other software processes that connect to a collective controller to query information about the collective and its membership or perform operations. Different client types are possible, including scripting languages, Java™, and JConsole. An admin client can connect to any collective controller in the collective and receive the same services; all collective controllers in the collective are equivalent. The collective supports an arbitrary number of concurrent admin clients.

Figure 1 illustrates the organizational concepts of a Liberty collective.

**Figure 1. Liberty collective organization**

A collective must have at least one controller and can have multiple controllers. When there are multiple controllers, they function as peers and replicate information with one another. A set of collective controllers is called a **replica set**. There is only one replica set per collective, and all controllers belonging to the same collective are part of the replica set. Replicas in the replica set communicate with one another through a dedicated replication port, protected by authentication and SSL. A collective with a single controller has a replica set of one. A replica set with three or more controllers provides the basis for scalability and high availability of the administrative domain. Three — not two — controllers are required to provide high availability due to use of a quorum algorithm. Increasing the size of a replica set enables more members in the collective. Additionally, a replica set offers multiple peer control points for the collective, providing for administrative client scaling, connection balancing, and failover.

A collective member is configured to connect to the collective’s replica set. A collective member must be configured to connect to at least one collective controller; it can be configured to connect to more than one controller in the replica set. A collective member only communicates with one
collective controller at a time; however, a configuration with more than one collective controller provides failover and workload balancing. Member-to-controller communication is always in the form of MBean operations performed over the IBM JMX REST Connector. Communication between controllers and members is always authenticated and protected with SSL.

An admin client connects to the collective much the same way as a collective member. The difference is an admin client has strictly an on-demand relationship with the replica set, querying information or invoking operations subject to its purpose. In contrast, a collective member has a contractual relationship with the replica set by which it publishes updates as its state changes; additionally, the replica set monitors members for liveness in order to update member status to reflect unplanned termination.

**Characteristics of a Liberty collective**

A Liberty collective has a number of design characteristics that facilitate management of Liberty in or out of the cloud environment. These characteristics combine to deliver an indigenous systems management foundation that is standards-based, flexible, lightweight, and scalable, all key properties for cloud enablement:

- **Standards-based management API**
  All management capabilities are built on Java Management eXtensions (JMX) MBeans. By keeping all management operations exposed through a standards-based model, management operations can be performed by leveraging common industry skills and tools (such as JConsole and Jython), and also enables custom tools to be easily built on top of the provided management API.

- **Loosely-coupled**
  The configuration of the collective exploits the Liberty composable server model by allowing servers to "opt-in" to a collective by providing minimal configuration — enough to establish a secure connection to a controller. This loosely-coupled model means Liberty servers can be quickly and easily moved in and out of a collective.

- **Distributed configuration**
  In the collective model, each Liberty server within the collective continues to own its configuration. The collective controller, which is the administrative server of the topology, does not dictate configuration nor store a master copy of the configuration of any Liberty server (other than its own configuration). This ensures fidelity when a Liberty server is removed from a collective.

- **Distributed cache model**
  The collective controller acts as a configuration and operational state cache. Each Liberty server joined to the collective publishes information about its operational and configuration state to the controller. By doing so, the controller has a cache of the sparse configuration and state of each collective member. This enables fast query of information about the administrative domain and its membership.

- **Agent-less**
  There are no agents within the collective topology. This means there is less cost on system resources. Operations, such as server starts, are performed through direct operating system RPC mechanisms, such as SSH or Windows® native RPC.
• **Scalable, resilient admin domain**
  Liberty collectives are designed to provide scalable and resilient administration by supporting multiple controllers within a collective. As mentioned above, the set of collective controllers within a collective is called a replica set. The replica set enables the size of a collective to scale out, as well as to provide a highly available administrative server environment.

**Deployment models**

There are two models for deploying a Liberty Collective: **build-up** and **push-out**. In both models, you must first provide a Liberty controller replica set.

**Build-up model**

In the Build-up model, you join existing Liberty servers to a collective. You can conceptualize this approach as shown in Figure 2.

**Figure 2. Build-up model**

You first create a Liberty server (if it does not already exist) using the Liberty profile server command. You next join the Liberty server to the collective using the Liberty profile collective command. The collective command "join" option establishes a trust association between the controller and member so that secure connections can be established between the two. The join option outputs a configuration snippet that must be added to the member's server.xml configuration file:

**Listing 1. Build-up model configuration**

```xml
<featureManager>
  <feature>collectiveMember-1.0</feature>
</featureManager>

<!-- controller connection and security configuration follows ... -->
```

Once the member configuration is updated, you can start it and it will join the collective and publish its essential state to the controller replica set.
See Related topics for information on establishing a replica set and details on how to apply the Build-up model.

**Push-out model**

In the Push-out model, you remotely add pre-packaged Liberty servers to a collective. You can conceptualize this approach as shown in Figure 3.

**Figure 3. Push-out model**

You first create and configure your Liberty server. This can be done either manually, using the Liberty Profile server command line interface and hand-editing the Liberty server configuration file (server.xml), or through WebSphere Developer Tools. Once the Liberty server is defined, you can package it into an archive using the Liberty profile server command "package" option.

Next, you deploy the archive using an admin client through a collective controller onto a collective host. This action both expands the archive and joins the member to the replica set. At this point, the new collective member is ready to run. You can start it and it will connect to the replica set and publish its state.

See Related topics for more information on using the push-out model.

**Operations**

Operations, including query, are performed by admin clients connected to the collective controller replica set. The connection is JMX over REST/HTTPS. All operations are provided by MBeans. The admin client can invoke any MBeans provided directly by the collective controller. The collective controller also functions as an MBean gateway to the collective members. By setting routing context, the admin client can also invoke MBeans on any of the collective members themselves. This enables an admin client to have access to any collective member over a single authenticated, secure connection to a collective controller.

Figure 4 illustrates operation routing.
Figure 4. Liberty collective operations routing

Configuration management

Configuration management is performed by modifying a Liberty server's configuration file (server.xml). The collective controller provides a file transfer operations through a FileTransfer MBean, which enables remote access to both collective controller and collective member configuration files. File download and upload operations can be invoked against either the collective controller or any collective member.

Figure 5 illustrates configuration management.

Figure 5. Liberty collective configuration management

Application scale and high availability

The Liberty collective supports lightweight clustering for Liberty servers. A collective member is not automatically a part of a Liberty cluster, but a Liberty server must be a collective member to
be part of a Liberty cluster. All Liberty servers in the same cluster must be part of the same Liberty collective.

The Liberty cluster design enables you to designate which servers are part of a named cluster. Liberty servers opt-in to clustering. You simply add the clusterMember feature and a cluster name to your server.xml file configuration. When the Liberty server starts up, it publishes this information to the collective's replica set and is then known as part of the named cluster. Liberty clusters can be managed using the ClusterManager MBean.

**Listing 2. Add Liberty cluster to main cluster**

```xml
<featureManager>
  <feature>clusterMember-1.0</feature>
</featureManager>
<clusterMember clusterName="MyAppCluster1" />
```

See [Related topics](#) for more information about configuring a Liberty cluster.

Figure 6 illustrates Liberty cluster support.

**Figure 6. Liberty cluster support**

The ClusterManager MBean enables you to perform a set of useful functions on a cluster:

- List clusters / list cluster members.
- Get cluster status.
- Start / stop cluster.
- Generated merged HTTP plug-in config.

The full API documentation can be found in the `com.ibm.websphere.appserver.api.collectiveController_1.0.0-javadoc.zip` file within the Liberty installation under dev/api/ibm/javadoc.

**Conclusion**

The Liberty collective provides a lightweight, scalable, highly available cloud-enabled systems management foundation for the Liberty Profile. It provides significant benefits to a multi-server
administrative domain that can greatly simplify deployment, operations, security management, and inventory awareness.

The Liberty collective is the foundation for new Liberty multi-server deployments and should be seriously considered as an integral part of any production deployment targeting the Liberty profile.
Related topics

- Introducing the Liberty Profile
- The Liberty Profile in 1 Minute
- Liberty Profile: Zero to Hero in Three and a-half Minutes
- Introducing Collectives
- Principles of the Liberty collective
- Video: Introduction to Creating a Collective
- Introducing Liberty Clusters
- Lab: Build and Administer Your Own Liberty Application Cluster
- WASdev: Community and resources for IBM WebSphere Application Server V8.5.5 Liberty profile
- Configuring a Liberty collective
- Liberty profile: Example of setting up a JMX routing environment

© Copyright IBM Corporation 2013
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)