High Availability of Zowe (Draft)

Last modified: Jul 14, 2020

Items marked PENDING requires more works, either investigation or implementation. The pending works should be specific about what is pending.

Similar to other services, availability of Zowe describes the period of time when Zowe service is available, as well as the time required by Zowe service to respond a request made by a user. By adding redundancies, clustering and failover mechanisms, Zowe provides infrastructure designs to decrease downtime, eliminating single point of failure and recover from failures. Some of the designs are built into Zowe components and some of the designs are reflected as configuration and administration guidance.

In below sections,

* **service** is mostly referring to Zowe service as whole,
* **component** is referring to Zowe sub-component running under Zowe service. This could be API Mediation Layer Gateway, Explorer APIs, Zowe App Framework (Desktop), or UI App running under Zowe App Framework, etc.
* **instance** is mostly referring to the particular process that Zowe component runs on the system.

## Introduction of The Solutions

Base on the environment you are targeting to run Zowe, we provide various solutions to achieve high availability. This section explains the solutions from architectural view and provides general ideas how Zowe works regarding High Availability.

### a). Stateful Component

In a high availability environment, Zowe has potential to start multiple instances of same component. For example, we may have two Zowe explorer dataset API instances running under same Zowe service for these purposes:

* one instance is working as a hot backup of the other one to increase available time,
* two instances both serve equally to the API consumers to increase the response time.

To support large range of load balancer, workload affinity is not supported. Zowe component should be stateless, so the workload can be directed to any active instance equally.

If one component stores session data inside instance and requires session affinity, in another word, this component is stateful, we encourage the component owner to convert the component to be stateless. This can be done by utilizing caching API provided by Zowe API Mediation layer. Any session related data should be stored in the caching service instead of the instance memory. The caching API supports several caching mechanisms and allows the system programmer to choose from based on their environment. This is introduced in [Choose and Configure A Caching Service](#choose-and-configure-a-caching-service) section.

PENDING:

1. implement and expose caching API from APIML,
2. the caching API should be resilient. Any downtime of this service will introduce downtime of all other components.

### b). Connectivity

The goal of connectivity high availability is for end-user, he can always connect to a healthy Zowe service and component instances, no matter there are failed instances behind the scene. Zowe takes advantage of several TCP/IP technologies to achieve high availability on connectivity, such as load balancing, port sharing, VIPA, Dynamic VIPA, Distributed DVIPA, etc. Load balancing is one key feature which directs the workload to active, healthy instance, and take the problematic instance out of scope.

**If you are running Zowe in single z/OS system**, with build-in load balancer, Ribbon, and proper configuration of port sharing or VIPA, your Zowe service will be resilient on connectivity. APIML Gateway should be configured with port sharing or VIPA, and all other components instances are under the gateway and traffic will be routed by APIML Ribbon load balancer. For each Zowe component, you have the choice of starting one or many instances to increase availability. When we start multiple instances of same component, the port listening on should be dynamic generated and the instance should be dynamically registered to APIML Discovery service.

PENDING: evaluate if this is our suggested solution. Test if this solution is compatible with WebSocket connections.

Detail configuration guidance can be found in [Configure Zowe for High Availability]() - [Running Zowe On A Single z/OS System]() section.

**If you are running Zowe in Parallel Sysplex environment**, the solution is combined with several configurations:

* Sysplex with shared ESM security domain. This is mandatory based on the fact of Zowe requires authentication service from z/OSMF.
* Sysplex Storage. This item suggests what Zowe volume, either runtime or instance configuration and workspace can be shared across multiple LPARs. And which part of the configuration has to be kept LPAR specific. Multiple Discovery Service should be aware of each other by distributing the Discovery Service config across Zowe service.
* Sysplex Distributor
* Port Sharing
* DVIPA and DDVIPA
* Workload Management
* z/OSMF on Sysplex

Similar to run Zowe in a single LPAR, you can still use Port Sharing to load balance traffic to APIML Gateway. You can also choose to use Sysplex Distributor / DVIPA to load balancing your multiple instances of Gateway.

For other component instances, there are two choices:

1. Start instances on dynamic ports and register the instance dynamically to Discovery service. This allows APIML Gateway to proxy the traffic to the right instance.
2. Manually configure port sharing / DVIPA for the component and let Gateway simply redirect traffic to your static component port.

PENDING:

1. decide which method is preferred.
2. When we use Port Sharing/DVIPA/Sysplex Distributor, is WebSocket connection fully supported.
3. Determine which part of Zowe file system can be shared across LPARs, and which part should not.
4. Determine mechanism how Discovery Service should be aware of each other.

 Detail configuration guidance can be found in [Configure Zowe for High Availability]() - [Running Zowe on z/OS Parallel Sysplex]() section.

**If you are running Zowe in docker container(s)**, the high availability of connectivity is provided by the container orchestrator, like Kubernetes or OpenShift. Running Zowe containers on other technologies, such as Docker Swarm, docker compose, are not suggested for production, high availability environment.

PENDING: consider ZSS resilience because it has to be run on z/OS even all other components are running in containers.

### C). Failover

This section defines how to identify service and/or instance failure, and who is responsible to recover from the failure. We provide different ways to achieve failover.

**If you are running Zowe in single z/OS system**, the failover of Gateway is automated by Port Sharing or VIPA. A failed Gateway instance will be taken over by the healthy standby immediately.

Failover of component instance is automatically handled by Zowe Lifecycle Facility. ZLF will find out if an instance is not available and determine if we need to start a new instance to replace it. The failover will stop when it hits predefined threshold. ZLF will write standard information to Zowe job log to help on identify the issues.

PENDING: determine which is better way to implement Zowe Lifecycle Facility. Few options:

1. Enhanced APIML Discovery Service - this enhanced version will perform active health check to identify failure and start new instance when it’s necessary. Question, how to make sure Discovery service is HA-enabled.
2. Independent zLaunch - this facility has registered information of how to start new instance of a component and will monitor processes / address spaces to determine if an instance is not available. Question, what if process still exists, but port binding is failing?
3. Use existing tools, adding configuration requirement:
	1. Automatic Restart Manager (ARM)
	2. System Automation
	3. Omegamon
	4. BMC Mainview

Questions:

1. can the ZLF handle service downgrade, for example, longer latency comparing to normal.
2. Will ZLF be smart enough to know the failure of starting new instance is caused by un-recoverable reasons and stop retrying: like resource shortage, permission denied, application flaw, etc.
3. Security concerns to avoid other process to highjack the failover protocol

Detail configuration guidance can be found in [Configure Zowe for High Availability]() - [Running Zowe On A Single z/OS System]() section.

**If you are running Zowe in Parallel Sysplex environment**, the solution is combined with configurations of:

* System Automation Policy

Similar to run Zowe in a single LPAR, you can still use Port Sharing or DVIPA to abandon a failed Gateway and switch to healthy Gateway automatically. You can also choose to import default System Automation Policy or customize it to determine and recover Gateway failure.

For other component instances, there are two choices:

1. Use the same Zowe Lifecycle Facility defined in running Zowe on single z/OS system.
2. Define more component level System Automation policies.

PENDING:

1. What’s the preferred method to failover on APIML Gateway.
2. Decide which method is preferred to failover instance.
3. Other than System Automation, are Automatic Restart Manger (ARM), Omegamon or BMC Mainview considered as alternatives?

 Detail configuration guidance can be found in [Configure Zowe for High Availability]() - [Running Zowe on z/OS Parallel Sysplex]() section.

**If you are running Zowe in docker container(s)**, the failover is provided by the container orchestrator, like Kubernetes or OpenShift.

### d). High Scalability

PENDING: if HA is in current scope. HS requires more information to determine how to scale:

1. Realtime workload of the instance, this will help the HS facility to decide when we need to scale,
2. Realtime LPAR resources for scaling, this will help the HS facility to decide where to scale.

## Configure Zowe for High Availability

The Zowe runtime and installation process may cover default or recommended configuration suggestions, but you may have to customize the configuration to fully fit in your workload expectation and z/OS system.

### a). Estimate Workload

To determine how many instances, LPARs you need for your Zowe service, you will need to estimate your workload. For example, if you Zowe users are heavy Desktop users, you may want to configure more Desktop instances than others.

### b). Running Zowe On A Single z/OS System

PENDING: based on the solution choice, define configuration guidance.

### c). Running Zowe on z/OS Parallel Sysplex

PENDING: based on the solution choice, define configuration guidance.

### d). Running Zowe In Docker Container(s)

High availability when you run Zowe in docker containers is provided by Kubernetes or OpenShift depends on how you start your containers. When you start Zowe containers in docker-compose, Docker Swarm or other means, the high availability could be limited due to the fact of technology. Please refer to the Zowe Containerization section for details how to start Zowe in Kubernetes or OpenShift.

## Configure Other Related Services to Enhance Zowe High Availability

### a). Configure High Availability for z/OSMF

Zowe relies on z/OSMF to provide authentication and also some Zowe components consumes other z/OSMF RESTful APIs. High availability of z/OSMF is one critical part of overall Zowe High Availability.

If you run Zowe on a single z/OS system, z/OSMF can be configured to enable hot or cold backup. Please find the solution described here. \*PENDING: find links to z/OSMF documentation\*

If you run Zowe on z/OS Parallel Sysplex, z/OSMF can also be configured in Sysplex environment. Please find more details here: [Configuring z/OSMF for high availability](https://www.ibm.com/support/knowledgecenter/SSLTBW\_2.3.0/com.ibm.zos.v2r3.izua300/izuconfig\_HighAvailabilityConfiguration.htm).

### b). Choose and Configure A Caching Service

Zowe requires a caching service to share stateful data across components. This stateful data may be some session-based

1. File System
2. VSAM Data Set
3. RabbitMQ
4. ActiveMQ
5. DB2
6. Redis
7. Memcached

PENDING:

1. if Caching Service, or Caching API is the right term,
2. determine which we want to support,
3. determine priority of implementation,
4. resolve security concerns
5. Caching API applies both z/OS and Containerized
6. Performance concerns

## Administration and Operation Guidance

### a). Identify Service Interruption or Downgrade

Zowe will write various messages into job log to help you to monitor Zowe availability.

APIML Discovery Service will write instance availability logs to Zowe job log. These are typical messages:

* ZWE????: component A instance #2 is not accessible
* ZWE????: component A instance #2 is recycled
* ZWE????: new instance #3 of component A is started on LPAR X
* ZWE????: starting new instance of component A failed due to ABCD, retrying 2 of 10 in 5 seconds

PENDING:

* revisit Zowe log format and validate if they meet requirement of System Automation
* Define and implement new messages.
* Add instance information to regular logs.

### b). Backup and Recover

Although with proper configured, Zowe can provide automated recovery from many failures, there are still chances some failure cannot be automated recovered and require manual intervene. This usually happens when we are running short of resources. This section describes what we should backup for recovery and how to recover from a failed Zowe instance.

### c). Rolling Update

During Zowe service lifetime, there are chances you will want to upgrade your Zowe to a higher version or apply PTFs. This is not only to apply latest features added into Zowe, it’s also a recommended practice to apply latest security fixes. To really achieve zero-downtime, rolling update is suggested when you upgrade Zowe. When we do rolling update on a running Zowe service, we don’t shutdown the whole service to upgrade, but replacing the component instance one by one to eliminate downtime.

Upgrading Zowe running on Parallel Sysplex is slightly different comparing to upgrade Zowe running on Single z/OS system.

Rolling update on Zowe running in container is provide by Kubernetes and OpenShift. You can use Kubernetes command line interface to upgrade component one by one.

PENDING: provide rolling update tool and usage guidance.

## Extending Zowe With Inherited High Availability

When you are extending Zowe server functionality with plugins, no matter you are trying to register your component to Zowe API Mediation Layer, or to create a new UI plugin under Zowe App Framework, in this section, you can find the information to help you to take advantage of the high availability provided by Zowe. These requirements are also applied to components which shipped with Zowe official releases.

### a). Stateful Service

Zowe requires the component or plugin to be stateless. If you have to store session specific information, you can use Caching API provided by API Mediation Layer.

### b). Health Check Endpoint

Component should provide a health check endpoint when register on APIML Discovery Service. This helps Discovery service to identify the availability of the instance. This is essential for connectivity high availability and automated failover.

### c). Register your component instance under APIML Discovery Service

To inherit the high availability feature from Zowe service, the instance should be registered dynamically to APIML Discovery Service.

### d). Connectivity and Failover

As a service registered under API Mediation Layer, Zowe component lifecycle handles instance failover and enable connectivity high availability. No special implementation and configuration requirement apply to the component except for instance definition. The instance definition includes these entries:

* min-instances: defines minimal quantity of instances you wish to start for your component. This value should be determined by how you
* max-instances: defines maximum quantity of instances you wish to start for your component.
* preferred-lpars: a list of preferred LPARs this component should be used.
* other-lpars:

### e). Logging

PENDING: provide guidance on how to write proper logging information to reflect these considerations:

* The log message should follow Zowe message standard (links to the standard).
* the log message should be able to display which instance is outputting the message.

## VI. Others

### a). Connect Zowe CLI to a Sysplex enabled z/OSMF

PENDING: we assume this should not be a problem but need to confirm.

### b). Using Certificate Under HA Environment

PENDING: identify potential issues, including using Keyring.

### C). Validate and Test HA Solutions

PENDING: how to validate our HA solutions.

### D). Benchmark HA Solutions Comparing to Non-HA Environment

PENDING: metrics:

1. Time to failover
2. Latency differences
3. Idle resources
4. State-change resources