Replicating Data Across Sites with Infinispan 10.1

Table of Contents

Chapter 1. Infinispan Cross-Site Replication

Cross-site replication allows you to back up data from one Infinispan cluster to another.

1.1. Cross-Site Replication

Infinispan clusters running in different locations can discover and communicate with each other.

A site is a locally running Infinispan cluster. For demonstration purposes, this documentation illustrates sites as data centers in different geographic locations, as in the following diagram:
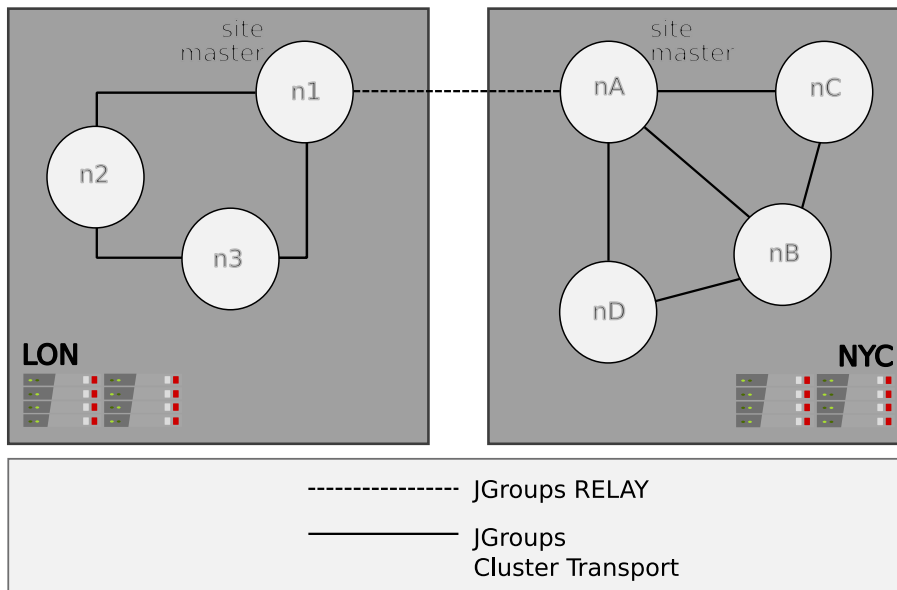
LON is a datacenter in London, England.
NYC is a datacenter in New York City, USA.

> Infinispan can form global clusters across two or more sites.
>
> For example, configure a third Infinispan cluster running in San Francisco, **SFO**, as backup location for **LON** and **NYC**.

1.1.1. Site Masters

Site masters are the nodes in Infinispan clusters that are responsible for sending and receiving requests from backup locations.

If a node is not a site master, it must forward backup requests to a local site master. Only site masters can send requests to backup locations.

For optimal performance, you should configure all nodes as site masters. This increases the speed of backup requests because each node in the cluster can backup to remote sites directly without having to forward backup requests to site masters.

1.2. Adding Backups to Caches

Name remote sites as backup locations in your cache definitions.

For example, the following diagram shows three caches, "customers", "eu-orders", and "us-orders":

- In **LON**, "customers" names **NYC** as a backup location.
- In **NYC**, "customers" names **LON** as a backup location.
- "eu-orders" and "us-orders" do not have backups and are local to the respective cluster.

1.3. Backup Strategies

Infinispan clusters can use different strategies for backing up data to remote sites.
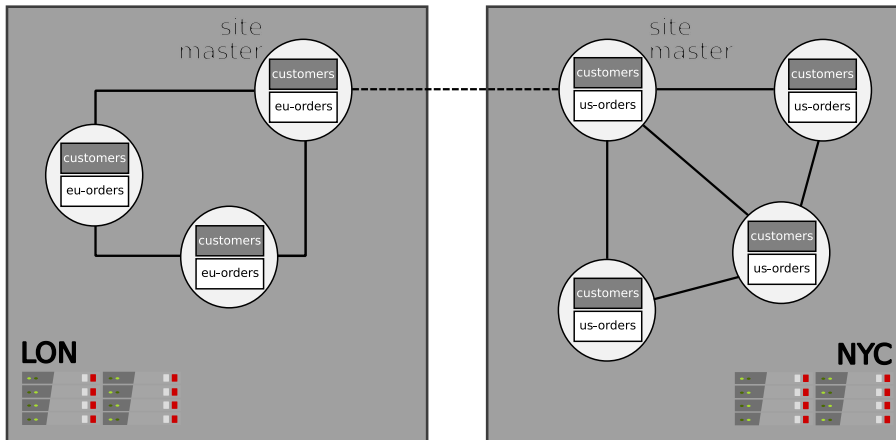
Infinispan replicates across sites at the same time that writes to local caches occur. For example, if a client writes "k1" to **LON**, Infinispan backs up "k1" to **NYC** at the same time.

1.3.1. Synchronous Backups

When Infinispan replicates data to backup locations, it waits until the operation completes before writing to the local cache.

You can control how Infinispan handles writes to the local cache if backup operations fail. For example, you can configure Infinispan to attempt to abort local writes and throw exceptions if backups to remote sites fail.

Synchronous backups also support two-phase commits with caches that participate in optimistic transactions. The first phase of the backup acquires a lock. The second phase commits the modification.

> Two-phase commit with cross-site replication has a significant performance impact because it requires two round-trips across the network.

1.3.2. Asynchronous Backups

When Infinispan replicates data to backup locations, it does not wait until the operation completes

before writing to the local cache.

Asynchronous backup operations and writes to the local cache are independent of each other. If backup operations fail, write operations to the local cache continue and no exceptions occur.

1.3.3. Synchronous vs Asynchronous Backups

Synchronous backups offer the strongest guarantee of data consistency across sites. If `strategy=sync`, when `cache.put()` calls return you know the value is up to date in the local cache and in the backup locations.

The trade-off for this consistency is performance. Synchronous backups have much greater latency in comparison to asynchronous backups.

Asynchronous backups, on the other hand, do not add latency to client requests so they have no performance impact. However, if `strategy=async`, when `cache.put()` calls return you cannot be sure of the value in the backup locations is the same as in the local cache.

1.4. Taking Sites Offline Automatically

Backup configurations include timeout values for operations to replicate data to remote sites. When backup operations reach the timeout, Infinispan records the operation as a failure.

To automatically take sites offline, you can then configure the number of **consecutive** failures that can occur.

For example, the **NYC** backup configuration specifies five as the number of failures after which **NYC** goes offline. If **LON** attempts five consecutive backup operations that fail, Infinispan automatically takes **NYC** offline. **LON** then stops backing up to **NYC** until you bring the site back online.

```
<backup site="NYC" strategy="ASYNC">
    <take-offline after-failures="5"/>
</backup>
```

You can also specify the amount of time to wait before taking sites offline. When backup operations fail, Infinispan waits before taking sites offline. If a backup request succeeds before the wait time runs out, Infinispan does not take the site offline.

```
<backup site="NYC" strategy="ASYNC">
    <take-offline after-failures="5"
                  min-wait="10000"/>
</backup>
```

In the preceding example, if failures occur for 5 consecutive operations, Infinispan waits 10 seconds and, if no requests are successful within the 10 second wait time, Infinispan then takes **NYC** offline.

To use only a minimum wait time for automatically taking locations offline, set a negative or zero value for the `after-failures` attribute; for example:

```
<backup site="NYC" strategy="ASYNC">
    <take-offline after-failures="-1"
                  min-wait="10000"/>
</backup>
```

💡 You can manually take sites offline through the Infinispan command line interface or REST API.

Reference

[Performing Cross-Site Replication Operations with the CLI or REST API](#)

1.5. State Transfer

State transfer is an administrative operation that synchronizes data between sites.

For example, **LON** goes offline and **NYC** starts handling client requests. When you bring **LON** back online, the Infinispan cluster in **LON** does not have the same data as the cluster in **NYC**.

To ensure the data is consistent between **LON** and **NYC**, you can push state from **NYC** to **LON**.

- State transfer is bidirectional. For example, you can push state from **NYC** to **LON** or from **LON** to **NYC**.

- Pushing state to offline sites brings them back online.

- State transfer overwrites only data that exists on both sites, the originating site and the receiving site. Infinispan does not delete data.

  For example, "k2" exists on **LON** and **NYC**. "k2" is removed from **NYC** while **LON** is offline. When you bring **LON** back online, "k2" still exists at that location. If you push state from **NYC** to **LON**, the transfer does not affect "k2" on **LON**.

  💡 To ensure contents of the cache are identical after state transfer, remove all data from the cache on the receiving site before pushing state. Use the `clear()` method.

- State transfer does not overwrite updates to data that occur after you initiate the push.

  For example, "k1,v1" exists on **LON** and **NYC**. **LON** goes offline so you push state transfer to **LON** from **NYC**, which brings **LON** back online. Before state transfer completes, a client puts "k1,v2" on **LON**.

  In this case the state transfer from **NYC** does not overwrite "k1,v2" because that modification happened after you initiated the push.

Reference

- org.infinispan.Cache.clear()

- Clearing Caches with the CLI

> 💡 Run `help clearcache` from the CLI for command details and examples.

- Clearing Caches with the REST API

1.6. Client Connections Across Sites

Clients can write to Infinispan clusters in either an Active/Passive or Active/Active configuration.

Active/Passive

The following diagram illustrates Active/Passive where Infinispan handles client requests from one site only:
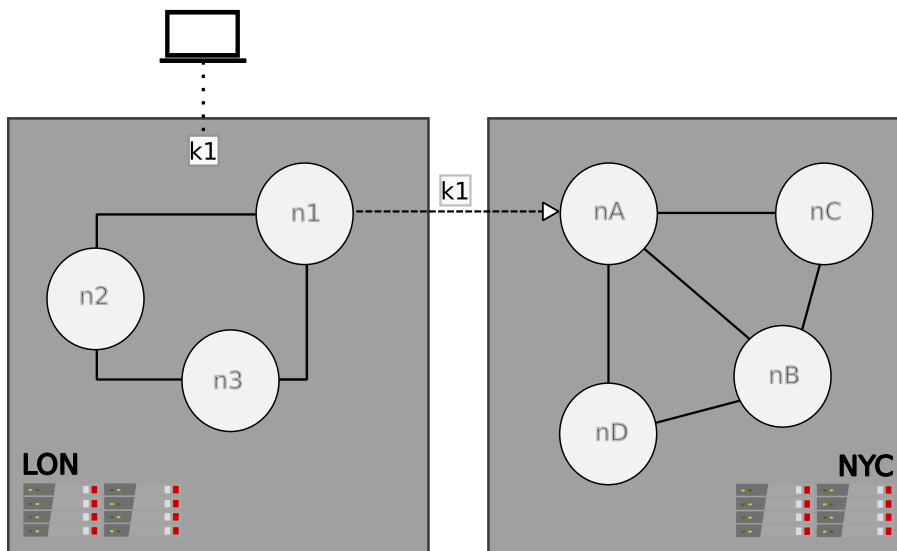
In the preceding image:

1. Client connects to the Infinispan cluster at **LON**.

2. Client writes "k1" to the cache.

3. The site master at **LON**, "n1", sends the request to replicate "k1" to the site master at **NYC**, "nA".

With Active/Passive, **NYC** provides data redundancy. If the Infinispan cluster at **LON** goes offline for any reason, clients can start sending requests to **NYC**. When you bring **LON** back online you can synchronize data with **NYC** and then switch clients back to **LON**.

Active/Active

The following diagram illustrates Active/Active where Infinispan handles client requests at two sites:
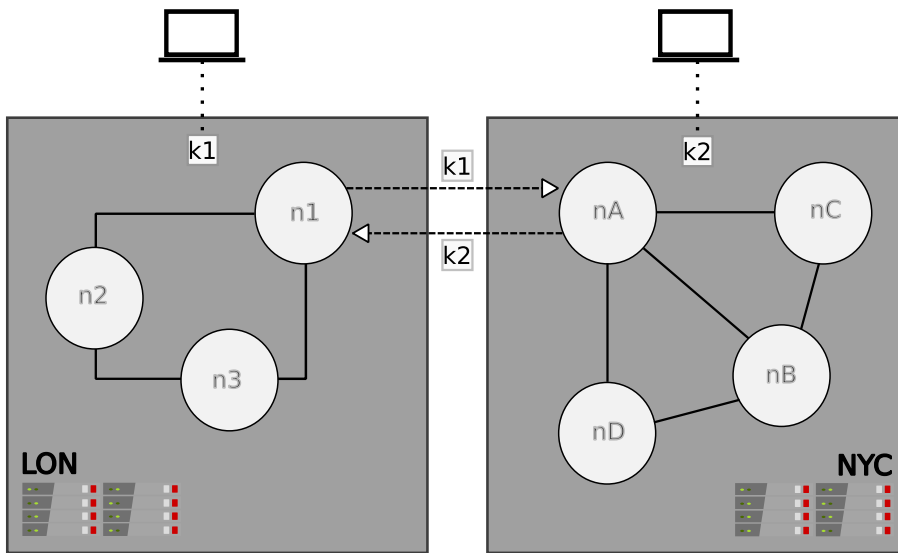
In the preceding image:

1. Client A connects to the Infinispan cluster at **LON**.

2. Client A writes "k1" to the cache.

3. Client B connects to the Infinispan cluster at **NYC**.

4. Client B writes "k2" to the cache.

5. Site masters at **LON** and **NYC** send requests so that "k1" is replicated to **NYC** and "k2" is replicated to **LON**.

With Active/Active both **NYC** and **LON** replicate data to remote caches while handling client requests. If either **NYC** or **LON** go offline, clients can start sending requests to the online site. You can then bring offline sites back online, push state to synchronize data, and switch clients as required.

1.6.1. Conflicting Entries with Cross-Site Replication

Conflicting entries can occur with Active/Active site configurations if clients write to the same entries at the same time but at different sites.

For example, client A writes to "k1" in **LON** at the same time that client B writes to "k1" in **NYC**. In this case, "k1" has a different value in **LON** than in **NYC**.

With synchronous replication, concurrent writes result in deadlocks because both sites lock the same key in different orders. To resolve deadlocks, client applications must wait until the locks time out.

With asynchronous replication, concurrent writes result in conflicting values because sites replicate after entries are modified locally. After replication occurs, there is no guarantee which value for "k1" exists at which site.

- Keys have conflicting values.

- One of the conflicting values is overwritten if sites do not replicate values at the same time. In

this case, one of the values is lost and there is no guarantee which value is saved.

In all cases, inconsistencies in key values are resolved after the next non-conflicting `put()` operation updates the value.

> There currently is no conflict resolution policy that client applications can use to handle conflicts in asynchronous mode. However, conflict resolution techniques are planned for a future Infinispan version.

1.7. Expiration and Cross-Site Replication

Infinispan expiration controls how long entries remain in the cache.

- `lifespan` expiration is suitable for cross-site replication. When entries reach the maximum lifespan, Infinispan expires them independently of the remote sites.
- `max-idle` expiration does not work with cross-site replication. Infinispan cannot determine when cache entries reach the idle timeout in remote sites.

Chapter 2. Configuring Infinispan for Cross-Site Replication

Configuring Infinispan to replicate data across sites, you first set up cluster transport so Infinispan clusters can discover each other and site masters can communicate. You then add backup locations to cache definitions in your Infinispan configuration.

2.1. Configuring Cluster Transport for Cross-Site Replication

Add JGroups RELAY2 to your transport layer so that Infinispan clusters can communicate with backup locations.

Procedure

1. Open `infinispan.xml` for editing.

2. Add the RELAY2 protocol to a JGroups stack, for example:

```
<jgroups>
    <stack name="xsite" extends="udp">
        <relay.RELAY2 site="LON" xmlns="urn:org:jgroups" max_site_masters="1000"/>
        <remote-sites default-stack="tcp">
            <remote-site name="LON"/>
            <remote-site name="NYC"/>
        </remote-sites>
    </stack>
</jgroups>
```

3. Configure Infinispan cluster transport to use the stack, as in the following example:

```
<cache-container name="default" statistics="true">
  <transport cluster="${cluster.name}" stack="xsite"/>
</cache-container>
```

4. Save and close `infinispan.xml`.

Reference

- JGroups RELAY2 Stacks
- Infinispan Configuration Schema

2.1.1. JGroups RELAY2 Stacks

Infinispan clusters use JGroups RELAY2 for inter-cluster discovery and communication.

```
<jgroups>
    <stack name="xsite" ①
           extends="udp"> ②
        <relay.RELAY2 xmlns="urn:org:jgroups" ③
                      site="LON" ④
                      max_site_masters="1000"/> ⑤
        <remote-sites default-stack="tcp"> ⑥
            <remote-site name="LON"/> ⑦
            <remote-site name="NYC"/>
        </remote-sites>
    </stack>
</jgroups>
```

① Defines a stack named "xsite" that declares which protocols to use for your Infinispan cluster transport.

② Uses the default JGroups UDP stack for intra-cluster traffic.

③ Adds `RELAY2` to the stack for inter-cluster transport.

④ Names the local site. Infinispan replicates data in caches from this site to backup locations.

⑤ Configures a maximum of 1000 site masters for the local cluster. You should set `max_site_masters` >= the number of nodes in the Infinispan cluster for optimal performance with backup requests.

⑥ Specifies all site names and uses the default JGroups TCP stack for inter-cluster transport.

⑦ Names each remote site as a backup location.

2.1.2. Custom JGroups RELAY2 Stacks

```
<jgroups>
    <stack-file name="relay-global" path="jgroups-relay.xml"/> ①
    <stack name="xsite" extends="udp">
        <relay.RELAY2 site="LON" xmlns="urn:org:jgroups"
                      max_site_masters="10" ②
                      can_become_site_master="true"/>
        <remote-sites default-stack="relay-global">
            <remote-site name="LON"/>
            <remote-site name="NYC"/>
        </remote-sites>
    </stack>
</jgroups>
```

① Adds a custom RELAY2 stack defined in `jgroups-relay.xml`.

② Sets the maximum number of site masters and optionally specifies additional RELAY2 properties. See JGroups RELAY2 documentation.

Example `jgroups-relay.xml`

```xml
<config xmlns="urn:org:jgroups"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:org:jgroups http://www.jgroups.org/schema/jgroups-
4.1.xsd">

    <!-- Use TCP for inter-cluster transport. -->
    <TCP bind_addr="127.0.0.1"
         bind_port="7200"
         port_range="30"

         thread_pool.min_threads="0"
         thread_pool.max_threads="8"
         thread_pool.keep_alive_time="5000"
    />

    <!-- Use TCPPING for inter-cluster discovery. -->
    <TCPPING timeout="3000"
             initial_hosts="127.0.0.1[7200]"
             port_range="3"
             ergonomics="false"/>

    <!-- Provide other configuration as required. -->
</config>
```

Reference

- Cluster Transport Configuration

- JGroups RELAY2

- Relaying between multiple sites (RELAY2)

2.2. Adding Backup Locations to Caches

Specify the names of remote sites so Infinispan can back up data to those locations.

Procedure

1. Add the `backups` element to your cache definition.

2. Specify the name of each remote site with the `backup` element.

   As an example, in the **LON** configuration, specify **NYC** as the remote site.

3. Repeat the preceding steps so that each site is a backup for all other sites. For example, you cannot add **LON** as a backup for **NYC** without adding **NYC** as a backup for **LON**.

   > Cache configurations can be different across sites and use different backup strategies. Infinispan replicates data based on cache names.

Example "customers" configuration in **LON**

```
<replicated-cache name="customers">
  <backups>
    <backup site="NYC" strategy="ASYNC" />
  </backups>
</replicated-cache>
```

Example "customers" configuration in **NYC**

```
<distributed-cache name="customers">
  <backups>
    <backup site="LON" strategy="SYNC" />
  </backups>
</replicated-cache>
```

Reference

- Infinispan Configuration Schema

2.3. Backing Up to Caches with Different Names

By default, Infinispan replicates data between caches that have the same name.

Procedure

- Use `backup-for` to replicate data from a remote site into a cache with a different name on the local site.

For example, the following configuration backs up the "customers" cache on **LON** to the "eu-customers" cache on **NYC**.

```
<distributed-cache name="eu-customers">
  <backups>
    <backup site="LON" strategy="SYNC" />
  </backups>
  <backup-for remote-cache="customers" remote-site="LON" />
</replicated-cache>
```

2.4. Verifying Cross-Site Views

After you configure Infinispan for cross-site replication, you should verify that Infinispan clusters successfully form cross-site views.

Procedure

- Check log messages for `ISPN000439: Received new x-site view` messages.

For example, if the Infinispan cluster in **LON** has formed a cross-site view with the Infinispan

cluster in **NYC**, it provides the following messages:

```
INFO  [org.infinispan.XSITE] (jgroups-5,${server.hostname}) ISPN000439: Received new
x-site view: [NYC]
INFO  [org.infinispan.XSITE] (jgroups-7,${server.hostname}) ISPN000439: Received new
x-site view: [NYC, LON]
```

2.5. Configuring Hot Rod Clients for Cross-Site Replication

Configure Hot Rod clients to use Infinispan clusters at different sites.

`hotrod-client.properties`

```
# Servers at the active site
infinispan.client.hotrod.server_list = LON_host1:11222,LON_host2:11222,LON_host3:11222

# Servers at the backup site
infinispan.client.hotrod.cluster.NYC =
NYC_hostA:11222,NYC_hostB:11222,NYC_hostC:11222,NYC_hostD:11222
```

`ConfigurationBuilder`

```
ConfigurationBuilder builder = new ConfigurationBuilder();
builder.addServers("LON_host1:11222;LON_host2:11222;LON_host3:11222")
       .addCluster("NYC")
       .addClusterNodes(
"NYC_hostA:11222;NYC_hostB:11222;NYC_hostC:11222;NYC_hostD:11222")
```

> 💡 Use the following methods to switch Hot Rod clients to the default cluster or to a cluster at a different site:
>
> - `RemoteCacheManager.switchToDefaultCluster()`
> - `RemoteCacheManager.switchToCluster(${site.name})`

Reference

- org.infinispan.client.hotrod.configuration package description
- org.infinispan.client.hotrod.configuration.ConfigurationBuilder
- org.infinispan.client.hotrod.RemoteCacheManager

Chapter 3. Performing Cross-Site Replication Operations

Bring sites online and offline. Transfer cache state to remote sites.

3.1. Cross-Site Operations with the CLI

The Infinispan command line interface lets you remotely connect to Infinispan servers, manage sites, and push state transfer to backup locations.

Prerequisites

- Start the Infinispan CLI.

- Connect to a running Infinispan cluster.

3.1.1. Bringing Backup Locations Offline and Online

Take backup locations offline manually and bring them back online.

Procedure

- Check if backup locations are online or offline with the `site status` command:

  ```
  //containers/default]> site status --cache=cacheName --site=NYC
  ```

  > ℹ️   `--site` is an optional argument. If not set, the CLI returns all backup locations.

- Bring backup locations online with the `bring-online` command:

  ```
  //containers/default]> site bring-online --cache=customers --site=NYC
  ```

- Take backup locations offline with the `take-offline` command:

  ```
  //containers/default]> site take-offline --cache=customers --site=NYC
  ```

For more information and examples, run the `help site` command.

3.1.2. Pushing State to Backup Locations

Transfer cache state to remote backup locations.

Procedure

- Use the `site` command to push state transfer, as in the following example:

```
//containers/default]> site push-site-state --cache=cacheName --site=NYC
```

For more information and examples, run the `help site` command.

Reference

Infinispan Command Line Interface

3.2. Cross-Site Operations with the REST API

Infinispan servers provide a REST API that allows you to perform cross-site operations.

3.2.1. Getting Status of All Backup Locations

Retrieve the status of all backup locations with `GET` requests.

```
GET /v2/caches/{cacheName}/x-site/backups/
```

Infinispan responds with the status of each backup location in JSON format, as in the following example:

```
{
  "NYC": "online",
  "LON": "offline"
}
```

Table 1. Returned Status

Value	Description
online	All nodes in the local cluster have a cross-site view with the backup location.
offline	No nodes in the local cluster have a cross-site view with the backup location.
mixed	Some nodes in the local cluster have a cross-site view with the backup location, other nodes in the local cluster do not have a cross-site view. The response indicates status for each node.

3.2.2. Getting Status of Specific Backup Locations

Retrieve the status of a backup location with `GET` requests.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}
```

Infinispan responds with the status of each node in the site in JSON format, as in the following example:

```
{
  "NodeA":"offline",
  "NodeB":"online"
}
```

Table 2. Returned Status

Value	Description
online	The node is online.
offline	The node is offline.
failed	Not possible to retrieve status. The remote cache could be shutting down or a network error occurred during the request.

3.2.3. Taking Backup Locations Offline

Take backup locations offline with GET requests and the ?action=take-offline parameter.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}?action=take-offline
```

3.2.4. Bringing Backup Locations Online

Bring backup locations online with the ?action=bring-online parameter.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}?action=bring-online
```

3.2.5. Pushing State to Backup Locations

Push cache state to a backup location with the ?action=start-push-state parameter.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}?action=start-push-state
```

3.2.6. Canceling State Transfer

Cancel state transfer operations with the ?action=cancel-push-state parameter.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}?action=cancel-push-state
```

3.2.7. Getting State Transfer Status

Retrieve status of state transfer operations with the `?action=push-state-status` parameter.

```
GET /v2/caches/{cacheName}/x-site/backups?action=push-state-status
```

Infinispan responds with the status of state transfer for each backup location in JSON format, as in the following example:

```
{
    "NYC":"CANCELED",
    "LON":"OK"
}
```

Table 3. Returned Status

Value	Description
SENDING	State transfer to the backup location is in progress.
OK	State transfer completed successfully.
ERROR	An error occurred with state transfer. Check log files.
CANCELLING	State transfer cancellation is in progress.

3.2.8. Clearing State Transfer Status

Clear state transfer status for sending sites with the `?action=clear-push-state-status` parameter.

```
GET /v2/caches/{cacheName}/x-site/local?action=clear-push-state-status
```

3.2.9. Modifying Take Offline Conditions

Sites go offline if certain conditions are met. Modify the take offline parameters to control when backup locations automatically go offline.

Procedure

1. Check configured take offline parameters with `GET` requests and the `take-offline-config` parameter.

   ```
   GET /v2/caches/{cacheName}/x-site/backups/{siteName}/take-offline-config
   ```

   The Infinispan response includes `after_failures` and `min_wait` fields as follows:

```
{
    "after_failures": 2,
    "min_wait": 1000
}
```

2. Modify take offline parameters in the body of PUT requests.

```
PUT /v2/caches/{cacheName}/x-site/backups/{siteName}/take-offline-config
```

3.2.10. Canceling State Transfer from Receiving Sites

If the connection between two backup locations breaks, you can cancel state transfer on the site that is receiving the push.

Cancel state transfer from a remote site and keep the current state of the local cache with the ?action=cancel-receive-state parameter.

```
GET /v2/caches/{cacheName}/x-site/backups/{siteName}?action=cancel-receive-state
```

3.2.11. Getting Status of Backup Locations

Retrieve the status of all backup locations from Cache Managers with GET requests.

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/
```

Infinispan responds with status in JSON format, as in the following example:

```
{
    "SFO-3":{
        "status":"online"
    },
    "NYC-2":{
        "status":"mixed",
        "online":[
            "CACHE_1"
        ],
        "offline":[
            "CACHE_2"
        ]
    }
}
```

Table 4. Returned Status

Value	Description
online	All nodes in the local cluster have a cross-site view with the backup location.
offline	No nodes in the local cluster have a cross-site view with the backup location.
mixed	Some nodes in the local cluster have a cross-site view with the backup location, other nodes in the local cluster do not have a cross-site view. The response indicates status for each node.

3.2.12. Taking Backup Locations Offline

Take backup locations offline with the `?action=take-offline` parameter.

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=take-
offline
```

3.2.13. Bringing Backup Locations Online

Bring backup locations online with the `?action=bring-online` parameter.

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=bring-
online
```

3.2.14. Starting State Transfer

Push state of all caches to remote sites with the `?action=start-push-state` parameter.

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-site/backups/{siteName}?action=start-
push-state
```

3.2.15. Canceling State Transfer

Cancel ongoing state transfer operations with the `?action=cancel-push-state` parameter.

```
GET /rest/v2/cache-managers/{cacheManagerName}/x-
site/backups/{siteName}?action=cancel-push-state
```