Oracle® Cloud

Using Caches in Oracle Application Container Cloud Service

E83061-16
October 2018

Oracle Cloud Using Caches in Oracle Application Container Cloud Service,

E83061-16

Primary Author: Rebecca Parks

Contents

Preface

Using Caches in Oracle Application Container Cloud Service describes how to create caches and use them in applications.

Topics:

- Audience
- Documentation Accessibility
- Related Resources
- Conventions

Audience

Using Caches in Oracle Application Container Cloud Service is intended for Oracle Cloud account administrators who are responsible for managing caches and developers who need to use them in applications.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Resources

See these Oracle resources:

- Oracle Public Cloud

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Getting Started with Caches

This guide tells you what you need to know to use caches in Oracle Application Container Cloud Service applications.

Topics:

About Caches in Oracle Application Container Cloud Service

Oracle Application Container Cloud Service features clustered, scalable, in-memory caching with data backup.

You specify the data capacity of a cache service. The topology is determined and provisioned automatically. You can add capacity by adding instances.

Data is replicated among cluster members in the cache service so nothing is lost should a member fail. If a member fails, data is redistributed among the remaining members to ensure resiliency.

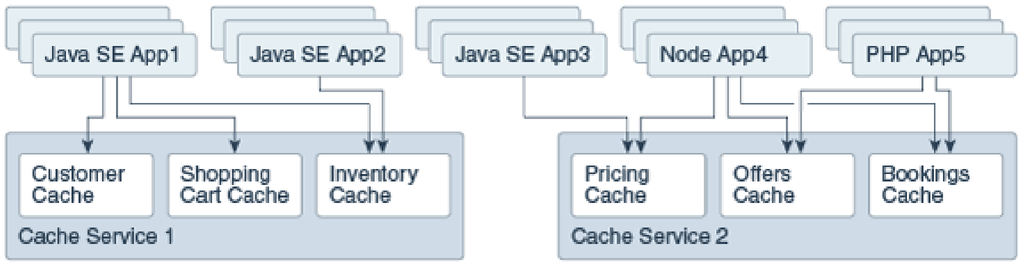
Common use cases for caches are:

- Reducing how often a data source is accessed, which results in better performance and scalability for applications.
- Sharing state information among multiple applications, which can be of different types.

Cache services, caches, and applications reference each other according to the following rules:

- One cache service can have multiple caches within it.
- One application can use multiple caches but only one cache service.
- Multiple applications can use the same cache service and caches.

The following diagram shows how each application can access multiple caches within one cache service, and how multiple applications can access the same cache service and share any caches within it.



Typical Workflow for Creating and Using Caches

To start using caches in Oracle Application Container Cloud Service, refer to the following tasks as a guide.

Task	Description	More Information
Learn about and use Oracle Application Container Cloud Service.	Before using caches, make sure you are familiar with other features of the service.	Typical Workflow for Administering Applications in *Using Oracle Application Container Cloud Service* and Typical Workflow for Developing Applications in *Developing for Oracle Application Container Cloud Service*
Access the Application Caches web user interface.	This is separate from the web user interface for applications.	Access Caches in Oracle Application Container Cloud Service
Create a cache service.	Create a cache service that applications can access.	Create a Cache Service or *REST API for Managing Application Caches*
Reference the cache URL in your application code.	Make sure your application reads the `CACHING_INTERNAL_CACHE_URL` environment variable at runtime.	The Cache URL Environment Variable
Use a Caching API in your application code.	You can use the Java API in a Java application. You can call the REST API from any language.	Use the Java API for Caching.
Reference a cache service when you deploy your application.	You can choose from a list of available cache services during deployment.	Reference a Cache During Application Deployment

Access Caches in Oracle Application Container Cloud Service

You access the Caching Service Console through the Oracle Application Container Cloud Service menu.

1. Open the service console for Oracle Application Container Cloud Service.
2. Click **Application Cache**.

   The Cache Services page of the Caching Service console appears. See Explore the Cache Services Page.

2

Creating and Managing Caches

This section describes how to create and manage caches in Oracle Application Container Cloud Service.

Topics:

- Explore the Cache Services Page
- Create a Cache Service
- Explore the Cache Service Overview Page

Explore the Cache Services Page

The Cache Services page lists cache services and lets you create, scale, restart, and delete them.

The following table describes the key information on the Cache Services page.

Element	Description
Search by service name	Filters the cache service list by name.
Create Service	Creates a new cache service. See Create a Cache Service.
Cache Service List	Displays the following information about each cache service. • Name — Click the name to display the cache service overview page. See Explore the Cache Service Overview Page. • Version — The cache service version. When Oracle Application Container Cloud Service is updated, existing caches remain at their initial version for backward compatibility. • Created On — Timestamp of cache service creation. • Memory — Amount of memory capacity allocated for the cache service. Click ☰ to perform the following actions. • Start — Start the cache service. • Stop — Stop the cache service. • Restart — Restart the cache service without data loss. • Scale Service — Change the memory or number of nodes for the cache service. • Delete — Delete the cache service.
Service Create and Delete History	Lists the creation and deletion history of all cache services.

Create a Cache Service

You can create a cache service from the Cache Services page.

See Access Caches in Oracle Application Container Cloud Service and Explore the Cache Services Page.

1. Open the Cache Services page.
2. Click **Create Service**.
3. On the Service page of the Create Service wizard, complete the fields described in this table:

Element	Description
Service Name	Enter the cache service name. The service name must start with a letter and must contain letters and numbers only. Spaces and other special characters are not allowed.
Service Description	(Optional) Enter a text description of the cache service.
Metering Frequency	Hourly is the only metering frequency currently supported.
Deployment Type	Select the deployment type: • **Basic** — Only one container is created for the cache service. • **Recommended** — Three or more containers are created for the cache service.
Cache Capacity [GB]	Enter the amount of data capacity to be allocated for the cache service.
Total Memory Allocated [GB]	Displays the total memory allocated for the cache service. This is greater than the cache capacity because the cache service holds multiple copies of the data.

4. Click **Next**.
5. Review the information on the Confirm page of the Create Service wizard.
6. Click **Create** to create the cache service.

   Click **Cancel** to cancel or **Previous** to go back to the Service page.

Explore the Cache Service Overview Page

The Cache Service Overview page lists information about a single cache service.

The following table describes the key information shown on the Cache Service Overview page.

Element	Description
Menu Icon	Click ☰ to perform the following actions. • Start — Start the cache service. • Stop — Stop the cache service. • Restart — Restart the cache service without data loss. • Scale Service — Change the memory or number of nodes for the cache service. • View Activity — Display the activity log.
Overview Tab	Displays the number of nodes for the cache service.
Service Overview	Displays the following information about the cache service. • Status — The cache service status. • Cache Capacity [GB] — Total amount of memory capacity allocated for the cache service. • Deployment Type — Either Basic or Recommended: – **Basic** — The cache service has only one container. – **Recommended** — The cache service has three or more containers. • Version — The cache service version. When Oracle Application Container Cloud Service is updated, existing caches remain at their initial version for backward compatibility. • Cache Host — The host name of the cache service.
Resources	Lists the caches within the cache service and the following information about each. • Container Name — Name of the cache. • Container Size — Total amount of memory capacity allocated for the cache.
Associations	Lists applications that use the cache service and the following information about each. • Service Name — Name of the application. • Type — Type of the application. • Status — Status of the application. Ready means it is running.
In-Progress Operation Messages	Lists operations that affect the cache service or applications that use it and the following information about each. Present only if such an operation is occurring. • Service Name — Name of the cache service or application. • Operation — Name of the operation. • Operation Status — Status of the operation. • Start Time — Timestamp of when the operation started. • End Time — Timestamp of when the operation ended.

3

Using Caching APIs in Applications

This section describes how to use the Java and REST APIs for caching applications to interact with Oracle Application Container Cloud Service caches.

Topics:

- The Cache URL Environment Variable
- Use the Java API for Caching
- Use the REST API in Applications
- Handle Connection Exceptions with Retries

The Cache URL Environment Variable

To access a cache service, your application must be able to read the `CACHING_INTERNAL_CACHE_URL` environment variable at runtime.

This variable contains effectively the hostname of the cache service. Your application uses this variable to construct the URL for communicating with the cache service.

The REST API for caching applications uses port `8080`.

Example 3-1 Reading the Environment Variable in a Java Application

```
public static final String CACHEHOST;
public static final String CACHEPORT = "8080";
public static final String CACHEURL;
static {
   CACHEHOST = System.getenv("CACHING_INTERNAL_CACHE_URL");
   if (CACHEHOST == null){
      System.out.println("CACHING_INTERNAL_CACHE_URL not set");
      System.exit(0);
   }
   CACHEURL = "http://" + CACHEHOST + ":" + CACHEPORT + "/ccs/";
}
```

Example 3-2 Reading the Environment Variable in a Node.js Application

```
process.on('exit', (code) => {
   console.log('CACHING_INTERNAL_CACHE_URL not set: ${code}');
});
var CCSHOST = process.env.CACHING_INTERNAL_CACHE_URL || process.exit(0);
var baseCCSURL = 'http://' + CCSHOST + ':8080/ccs/';
```

Example 3-3 Reading the Environment variable in a PHP Application

```
$cacheService = getenv('CACHING_INTERNAL_CACHE_URL') ?
getenv('CACHING_INTERNAL_CACHE_URL') : '';
if(strlen($cacheService) == 0)
{
    throw new Exception('No cache service found', 500);
}
$base_url = 'http://'.$cacheService.':8080/ccs/'
```

Use the Java API for Caching

The Java API for caching enables your Java application to write values to, read values from, delete values in, and clear the cache. This API is an open source framework with sources hosted on GitHub and binaries both directly downloadable from and available through Maven Central.

This section describes the conceptual framework and basic steps for how to use the Java API for caching. You can download the Java classes and read the Javadoc pages at Oracle Application Container Cloud Service Java SDK for Caching. Tutorials that use this API are Creating an Application Using the Java API for Caching on Oracle Application Container Cloud Service and Creating an Application Using the Java API for Caching and a Database on Oracle Application Container Cloud Service.

To access a cache, your application must first define a `SessionProvider` and a `Session` object:

1. Create a `SessionProvider` obejct for a cache service that you previously created. Provide a URL with the cache service name and the port that supports the desired transport protocol: `1444` for GRPC or `8080` for REST. When using REST, the hostname is followed by `/ccs`. A GRPC example:

   ```
   SessionProvider sp = new RemoteSessionProvider("http://MyCache:1444");
   ```

   A REST example:

   ```
   SessionProvider sp = new RemoteSessionProvider("http://MyCache/ccs:8080");
   ```

2. Obtain a `Session` object from the `SessionProvider` object using a specific transport. A GRPC example:

   ```
   Session cs = sp.createSession(Transport.grpc());
   ```

   A REST example:

   ```
   Session cs = sp.createSession(Transport.rest());
   ```

3. Obtain a `Cache` object from the `Session` object. The cache in this example is named `users`, and it stores objects of class `User`.

   ```
   Cache<Users> users = cs.getCache("users");
   ```

After your application has obtained a `Cache` object, it can interact with the cache. The API includes the methods `Cache.get()`, `Cache.put()`, `Cache.replace()`, and `Cache.remove()`. For example, the code to put a `User` object into the `users` cache with the user's ID as the key is as follows:

```
users.put(user.getId(),user);
```

Removing an object from a cache can be as simple as calling `users.remove(id)`, but because caches are not local, the remove operation provides optimization options. The `Cache.remove()` method lets you specify, among other things, whether you want the removed object returned. By default it returns `null`. To return the removed object, use the `Return.OLD_VALUE` option. For example:

```
User user = users.remove(id, Return.OLD_VALUE);
```

Like `Cache.remove()`, the `Cache.replace()` method provides optimization options and returns `null` by default. For example:

```
users.replace(id, user);
```

For full details about these methods, see the Javadoc pages.

Use the REST API in Applications

The REST API for caching applications enables your application to write values to, read values from, delete values in, and clear the contents of the cache. You can call this API from any language.

This API is fully described in *REST API for Using Caches in Applications*. The following tutorials show examples of applications in various languages:

- Creating a Java Application Using the Caching REST API in Oracle Application Container Cloud Service
- Creating a Node.js Application Using the Caching REST API in Oracle Application Container Cloud Service
- Create a Python Application Using the Caching REST API

Handle Connection Exceptions with Retries

Sometimes when a cache service is scaled in, scaled out, or restarted, it can't accept connections for a brief time. For a Java application that attempts to connect, a `java.netConnectionException` or `java.net.UnknownHostException` might occur. You can code a retry mechanism to handle these exceptions.

This retry mechanism should control the number of retries attempted before giving up and any thread sleep time between attempts.

The following is a simplified example of a retry mechanism written in Java to serve as a guide. It employs a helper class, `RetryOnException`.

```
/**
 * Encapsulates retry-on-exception operations
 */
public class RetryOnException {
    public static final int DEFAULT_RETRIES = 30;
    public static final long DEFAULT_TIME_TO_WAIT_MS = 2000;

    private int numRetries;
    private long timeToWaitMS;

    // CONSTRUCTORS
    public RetryOnException(int _numRetries,
                            long _timeToWaitMS)
    {
        numRetries = _numRetries;
        timeToWaitMS = _timeToWaitMS;
    }

    public RetryOnException()
    {
        this(DEFAULT_RETRIES, DEFAULT_TIME_TO_WAIT_MS);
    }
    /**
     * shouldRetry
     * Returns true if a retry can be attempted.
     * @return  True if retries attempts remain; else false
     */
    public boolean shouldRetry()
    {
        return (numRetries >= 0);
    }

    /**
     * waitUntilNextTry
     * Waits for timeToWaitMS. Ignores any interrupted exception
     */
    public void waitUntilNextTry()
    {
        try {
            Thread.sleep(timeToWaitMS);
        }
        catch (InterruptedException iex) { }
    }

    /**
     * exceptionOccurred
     * Call when an exception has occurred in the block. If the
     * retry limit is exceeded, throws an exception.
     * Else waits for the specified time.
     * @throws Exception
     */
    public void exceptionOccurred() throws Exception
    {
        numRetries--;
        if(!shouldRetry())
        {
            throw new Exception("Retry limit exceeded.");
        }
        waitUntilNextTry();
    }
}
```

Here is a Java method, `getWithRetries`, that illustrates how to wrap a REST `GET` operation in the retry handler. `CACHE_NAME` is a String variable that holds the name of the cache to access. You should tune the `retries` and `retrySleep` parameter values for your application. Testing suggests that connectivity exceptions occur in a window of about 5 seconds at most, so reasonable starting values for `retries` and `retrySleep` might be `20` and `500`, respectively.

```
    /**
     * getWithRetries
     * Issues a REST GET with retries. Throws Exception if the
     * GET could not succeed after retries attempts.
     * If the GET was successful, but returned anything other than an HTTP 200
     * status, return null;
     * If successful, returns the value.
     * @param target     WebTarget for the REST call
     * @param key        The cache key to fetch
     * @param retries    Number of retries to attempt
     * @param retrySleep Sleep time in milliseconds between each attempt
     * @return  String value, or null if the GET did not return an HTTP 200 code
     * @throws Exception
```

```
 */
public String getWithRetries(WebTarget target,
                             String key,
                             int retries,
                             long retrySleep) throws Exception
{
    Response getResponse = null;
    boolean success = false;
    int getStatus;

    // For handling retries
    RetryOnException retryHandler = new RetryOnException(retries, retrySleep);

    while(true) {
        try {
            getResponse = target
                    .path(CACHE_NAME + "/" + key)
                    .request(MediaType.APPLICATION_OCTET_STREAM)
                    .get();
        }
        // Catch exception and retry.
        // If beyond retry limit, this will throw an exception.
        catch (Exception ex)
        {
            retryHandler.exceptionOccurred();
            continue;
        }

        // If the status is not a 200, return a NULL.
        // Otherwise, exit the loop to return the value.
        getStatus = getResponse.getStatus();
        if(getStatus != 200)
        {
            return null;
        }
        else
        {
            break;
        }
    }

    // Return the result
    return getResponse.readEntity(String.class);
}
```

4

Referencing Caches in Deployed Applications

This section describes how to create service bindings to caches in Oracle Application Container Cloud Service.

Reference a Cache During Application Deployment

Specifying a cache service for your application during deployment is recommended.

Deploying an application to Oracle Application Container Cloud Service is fully described in Creating an Application in *Using Oracle Application Container Cloud Service*.

When you select a cache service during deployment, the corresponding `CACHING_INTERNAL_CACHE_URL` environment variable is automatically added to the Environment Variables list. See The Cache URL Environment Variable.

Reference a Cache After Application Deployment

To reference a cache service in an already application deployed, you use a service binding.

Service bindings are fully described in Managing Service Bindings in *Using Oracle Application Container Cloud Service*, and The Cache URL Environment Variable.