CloudMonitor

Developer Guide

Developer Guide

Welcome to Alibaba Cloud CloudMonitor. This guide provides detailed information about CloudMonitor API operations.

Currently, interfaces are provided for querying metric data and setting alarm rules. Before using these interfaces, ensure you have fully understood product instructions and user agreements.

Note

- OpenAPI provides metric data of the last 31 days.
- A maximum of 7,000 alarm rules can be set under each Alibaba Cloud account.

Request structure

Service address

Region	Service address
China East 1 (Hangzhou)	metrics.cn-hangzhou.aliyuncs.com
China East 2 (Shanghai)	metrics.cn-shanghai.aliyuncs.com
China North 1 (Qingdao)	metrics.cn-qingdao.aliyuncs.com
China North 2 (Beijing)	metrics.cn-beijing.aliyuncs.com
China South 1 (Shenzhen)	metrics.cn-shenzhen.aliyuncs.com
China North 3 (Zhangjiakou)	metrics.cn-zhangjiakou.aliyuncs.com
Hong Kong	metrics.cn-hongkong.aliyuncs.com
Singapore	metrics.ap-southeast-1.aliyuncs.com
US West 1 (Silicon Valley)	metrics.us-west-1.aliyuncs.com
US East 1 (Virginia)	metrics.us-east-1.aliyuncs.com
Germany 1 (Frankfurt)	metrics.eu-central-1.aliyuncs.com
Asia Pacific SE 2 (Sydney)	metrics.ap-southeast-2.aliyuncs.com
Middle East 1 (Dubai)	metrics.me-east-1.aliyuncs.com
Asia Pacific NE 1 (Japan)	metrics.cn-hangzhou.aliyuncs.com

Asia Pacific SE 3 (Kuala Lumpur)	metrics.ap-southeast-3.aliyuncs.com
China North 5 (Huhehaote)	metrics.cn-huhehaote.aliyuncs.com

Note: When you query the metric data of Asia Pacific NE 1 (Japan), use China East 1 (Hangzhou) service address.

Communication protocols

Request communication by using HTTP is supported.

Request methods

The system allows you to send HTTP GET or POST requests. In HTTP GET mode, request parameters must be included in the request URL.

Request parameters

Each request must contain the request parameters for public authentication and signatures and also the parameters for the related operations.

Character encoding

Requests and return results are encoded by using the UTF-8 character set.

Public parameters

Name	Type	Required	Description
Format	String	No	Type of returned value. Value range: JSON, XML. Default value: XML.
Version	String	Yes	API version number. Format YYYY-MM-DD. The current version is 2017-03-01.
AccessKeyId	String	Yes	Key ID that Alibaba Cloud issues to you to access services.
Signature	String	Yes	Signature result string. For more information about the signature calculation method,

			see Signature mechanism.
SignatureMethod	String	Yes	The signature method. HMAC-SHA1 is supported currently.
Timestamp	String	Yes	Timestamp of a request. The date format complies with ISO8601 and uses UTC time. Format: YYYY-MM-DDThh:mm:ssZ. For example, 2017-03-23T06:59:55Z (for 14:59:55 March 23, 2017 Beijing time).
SignatureVersion	String	Yes	Signature algorithm version. The current version is 1.0.
SignatureNonce	String	Yes	Unique random number, used to prevent replay attacks. You must use different random numbers for different requests.

Signature mechanism

Each time CloudMonitor receives an access request, it performs sender authentication. Therefore, the HTTP request must contain signature information.

By using AccessKey ID and AccessKey Secret, CloudMonitor performs symmetric encryption to authenticate the request sender.

The AccessKey ID and AccessKey Secret are issued to visitors by Alibaba Cloud (visitors can apply for and manage them at the website of Alibaba Cloud). The AccessKey ID indicates the identity of a visitor, and the AccessKey Secret is the key used to encrypt a signature string and verify it at the server end. It must be kept confidential and must only be available to Alibaba Cloud and the user.

Signature handling

Use request parameters to construct a canonicalized query string.

Follow the following rules to construct the string for signature calculation with the

canonicalized query string.

```
StringToSign=
HTTPMethod + "&" +
percentEncode("/") + "&" +
percentEncode(CanonicalizedQueryString)
```

Here, HTTPMethod is the HTTP method used for request submission. For example, GET.percentEncode(“/ ”) is the encoded value for the character “/ “ based on the URL encoding rules described in 1.b, that is, “%2F” .

percentEncode(CanonicalizedQueryString) is the string encoded from the canonicalized request string structured in step 1 according to the URL Coding Rule in 1.b.

Use the preceding signature string to calculate the HMAC value of the signature based on RFC2104 definitions.

> Note: The key used for signature calculation is your AccessKey Secret adding the ampersand (&) (ASCII:38). The key is based on hash algorithm SHA1.

Encode the HMAC value into a string based on Base64 encoding rules to obtain the signature value.

Add the obtained signature value to request parameters.

Request string constructing

The request parameters are ordered alphabetically by the parameter names (this includes the “public request parameters” and custom parameters for the given request interfaces described in this document, but not the Signature parameter mentioned in “Public request parameters”).

> Note: For a request submitted using the GET method, these parameters is part of the request URI (that is, the section in the URI following “?” and connected by “&”).

The name and value of each request parameter are encoded. The names and values must undergo URL encoding using the UTF-8 character set.

a. Uppercase letters from A to Z, lowercase letters from a to z, integers from 0 to 9, and other characters including en dashes (-), underlines (_), periods (.), and tildes (~) are not

encoded.

b. Other characters are encoded in (%XY) format, with XY representing the characters' ASCII code in hexadecimal notation. For example, double quotation marks (") are encoded as %22.

c. Extended UTF-8 characters are encoded in (%XY%ZA...) format.

d. It must be noted that an English space ( ) is encoded as %20, rather than the plus sign (+).

> Note: Generally, libraries that support URL encoding (for example, java.net.URLEncoder of Java) are all encoded according to the rules for the "application/x-www-form-urlencoded" MIME-type. You can use this encoding method directly by replacing the plus sign (+) in the encoded string with "%20", the asterisk (*) with (%2A), and change (%7E) back to the tilde ~) to conform to the encoding rules described earlier.

Connect the encoded parameter names and values with the equal sign (=).

Sort the parameter name and value pairs connected by equal signs in alphabetical order, and connect them with the & symbol to produce the Canonicalized Query String.

Take QueryMetricList as an example, the request URL before signature is:

```
http://metrics.aliyuncs.com/?Action=QueryMetricList&period=60&StartTime=2016-03-22T11:30:27Z&Dimensions={instanceId:&apos;i-abcdefgh123456&apos;}&Timestamp=2017-03-23T06:59:55Z&Project=acs_ecs_dashboard&SignatureVersion=1.0&Format=JSON&SignatureNonce=aeb03861-611f-43c6-9c07-b752fad3dc06&Version=2015-10-20&AccessKeyId=TestId&Metric=cpu_idle&SignatureMethod=HMAC-SHA1
```

The corresponding StringToSign is:

```
GET&%2F&AccessKeyId%3DTestId%26Action%3DQueryMetricList%26Dimensions%3D%257B%2522instanceId%2522%253A%2522i-abcdefgh123456%2522%257D%26Format%3DJSON%26Metric%3Dcpu_idle%26Period%3D60%26Project%3Dacs_ecs_dashboard%26SignatureMethod%3DHMAC-SHA1%26SignatureNonce%3Daeb03861-611f-43c6-9c07-b752fad3dc06%26SignatureVersion%3D1.0%26StartTime%3D2016-03-22T11%253A30%253A27Z%26Timestamp%3D2017-03-23T06%253A59%253A55Z%26Version%3D2015-10-20
```

Assume that the AccessKey ID parameter value is TestId, the AccessKey Secret parameter value is TestSecret, then the key used for HMAC calculation is TestSecret& and the calculated signature value is:

```
TLj49H/wqBWGJ7RK0r84SN5IDfM=
```

The signed request URL is (signature parameter is added):

```
http://metrics.cn-hangzhou.aliyuncs.com/?Action=QueryMetricList&StartTime=2016-03-22T11%3A30%3A27Z&Period=60&Dimensions=%7B%22instanceId%22%3A%22i-abcdefgh123456%22%7D&Timestamp=2017-03-23T06%3A59%3A55Z&Project=acs_ecs_dashboard&SignatureVersion=1.0&Format=JSON&SignatureNonce=aeb03861-611f-43c6-9c07-b752fad3dc06&Version=2015-10-20&AccessKeyId=TestId&Metric=cpu_idle&SignatureMethod=HMAC-SHA1&Signature=TLj49H%2FwqBWGJ7RK0r84SN5IDfM%3D
```

The way to call a CloudMonitor service interface is to send an HTTP request to the CloudMonitor server (only HTTP is supported currently) and receive the response.

After receiving a user request, the CloudMonitor server performs necessary authentication and parameter verification on the request. After all verifications are successful, the server submits or performs the associated operations based on the parameters specified for the request. Then, it returns the results to you in the form of an HTTP response.

Request composition

Requests are composed of the following parts:

- HTTP method: Currently, all CloudMonitor interfaces support GET method calls. Metric data can also be reported through POST.
- Request URL: The request URL contains the service address of a request, name of the operation to be executed, operation parameters, and public request parameters.
- Server address: The domain name of the CloudMonitor service is http://metrics.aliyuncs.com/ .
- Operation name: Each interface requires you to specify the name of the operation you want to execute, that is, the Action parameter.
- Operation parameter: Different operation parameters must be set for different operations to be executed.
- Public request parameter: Parameters required for each request, such as Time Stamp and Signature.
- To ensure that the server can verify the user’ s authentication and authorize the request execution, signature must be signed before the submission of the request. For signature rules, see the signature mechanism section in Call methods.
- After the server finishes processing the request, response results are returned. Response results are categorized into successful results and error messages. You can parse the response message body to obtain the execution results.

Request parameters

Name	Type	Required	Description

Action	String	Yes	Operation interface, required parameter. Value: QueryMetricList.
Project	String	Yes	Name space. It indicates the product to which the metric data belongs, for example, “acs_rds”. Namespaces can be used.
Metric	String	Yes	The metric item name. For available names, see the metric list.
Period	String	No	The time interval, calculated in seconds. For example, 60, 300, 900. If this is not entered, the raw data is queried based on the report period stated during metric item registration. If the statistical period is entered, the corresponding statistical data is queried.
StartTime	String	Yes	The start time. It can be a millisecond value counted from 00:00, January 1, 1970, or formatted data, such as 2015-10-20 00:00:00.
EndTime	String	No	It can be a millisecond value counted from 00:00, January 1, 1970, or formatted data, such as 2015-10-20 00:00:00.
Dimensions	String	Yes	It locates the dimension of the metric data position. In the example of the metric item “disk IO”, the unique monitoring position can be

			located via two dimensions, namely, instance and disk name.
Cursor	String	No	Paging cursor. It indicates that the next query continues with the last query.
Length	String	Yes	Number of items to be queried this time. The maximum value is 1,000. If set to any number greater than 1,000, this parameter is automatically reset to 1,000.

Parameter description

For details about how to assign values for Project, Metric, Period, and Dimensions, see Preset metric item reference.

startTime and endTime are in the left-open and right-closed mode, and startTime must be earlier than endTime.

Cursor is a parameter in paging mode. The Cursor parameter indicates there is a next page. If the return result is null, there are no more pages.

Generally, the Period value can be 60 (1 minute), 300 (5 minutes), or 900 (15 minutes). You can set the Period parameter based on the document requirements and query scenario. For example, to query data for a specified day, if you set Period to 60, then 1,000 data items are returned (actually there are 1,440 items but only 1,000 items are returned since the maximum return value cannot be greater than 1,000); if you set Period to 300, then 288 data items are returned.

The first letter of each parameter is in uppercase.

The interface supports RAM subaccount calls. The operation descriptor at authorization is “cms:QueryMetricList” and the resource descriptor is “*”.

Return parameters

Name	Type	Description
Code	String	Return code. It generally includes 200 (normal), 400 (parameter error), 403 (permission error), 500 (server error). Return codes except 200 are all error codes.
Msg	String	The status description returned by the query. Msg is blank when Code is “200”.
Success	Boolean	Whether or not the current query was performed successfully. If there is an exception on the server side, the returned value is “false” (and “true” otherwise).
Size	Integer	Number of actually returned items in the current query.
RequestId/TracerId	String	Unique request ID. When there is an issue with querying, you can provide this field to a technician to troubleshoot the problem.
Datapoints	JSON	Metric data.
Cursor	String	If the number of queried items is greater than the length value, Cursor is returned as a parameter for the next query. If Cursor is not returned, all items are queried.

Return value error code

Error Code	Description	Meaning
400	Bad Request	Parameter error.
403	Forbidden	Permission restriction. AccessKey does not match the item to be queried.
500	Internal Server Error	Server error.

Call example

Signature code:

UrlUtil.java

```
import java.net.URLEncoder;
import java.util.Map;

import org.apache.commons.lang.StringUtils;
import org.apache.log4j.Logger;

public class UrlUtil {
private static Logger logger = Logger.getLogger(UrlUtil.class);

private final static String CHARSET_UTF8 = "utf8";

/**
*
* @param url
* @return
*/
public static String urlEncode(String url) {
if (!StringUtils.isEmpty(url)) {
try {
url = URLEncoder.encode(url, "UTF-8");
} catch (Exception e) {
logger.warn("Url encode error:" + e.getMessage());
}
}

return url;
}

public static String generateQueryString(Map<String, String> params, boolean isEncodeKV) {
StringBuilder canonicalizedQueryString = new StringBuilder();
for (Map.Entry<String, String> entry : params.entrySet()) {
if (isEncodeKV)
canonicalizedQueryString.append(percentEncode(entry.getKey())).append("=")
.append(percentEncode(entry.getValue())).append("&");
else
canonicalizedQueryString.append(entry.getKey()).append("=")
.append(entry.getValue()).append("&");
}
if (canonicalizedQueryString.length() > 1) {
canonicalizedQueryString.setLength(canonicalizedQueryString.length() - 1);
}
return canonicalizedQueryString.toString();
}

public static String percentEncode(String value) {
try {
// After using URLEncoder.encode for encoding, replace"+","*", and "%7E" with values that conform to the
```

```
encoding standard specified by the API
return value == null ? null : URLEncoder.encode(value, CHARSET_UTF8)
.replace("+", "%20").replace("*", "%2A").replace("%7E", "~");
} catch (Exception e) {
//Impossible exception
}
return "";
}
}
```

SignatureUtils.java

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.Map;
import java.util.TreeMap;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang.StringUtils;


public class SignatureUtils {

private final static String CHARSET_UTF8 = "utf8";
private final static String ALGORITHM = "UTF-8";
private final static String SEPARATOR = "&";

public static Map<String, String> splitQueryString(String url)
throws URISyntaxException, UnsupportedEncodingException {
URI uri = new URI(url);
String query = uri.getQuery();
final String[] pairs = query.split("&");
TreeMap<String, String> queryMap = new TreeMap<String, String>();
for (String pair : pairs) {
final int idx = pair.indexOf("=");
final String key = idx > 0 ? pair.substring(0, idx) : pair;
if (!queryMap.containsKey(key)) {
queryMap.put(key, URLDecoder.decode(pair.substring(idx + 1), CHARSET_UTF8));
}
}
return queryMap;
}

public static String generate(String method, Map<String, String> parameter,
String accessKeySecret) throws Exception {
String signString = generateSignString(method, parameter);
System.out.println("signString---"+signString);
byte[] signBytes = hmacSHA1Signature(accessKeySecret + "&", signString);
```

```
String signature = newStringByBase64(signBytes);
System.out.println("signature---"+signature);
if ("POST".equals(method))
return signature;
return URLEncoder.encode(signature, "UTF-8");

}

public static String generateSignString(String httpMethod, Map<String, String> parameter)
throws IOException {
TreeMap<String, String> sortParameter = new TreeMap<String, String>();
sortParameter.putAll(parameter);

String canonicalizedQueryString = UrlUtil.generateQueryString(sortParameter, true);
if (null == httpMethod) {
throw new RuntimeException("httpMethod can not be empty");
}

StringBuilder stringToSign = new StringBuilder();
stringToSign.append(httpMethod).append(SEPARATOR);
stringToSign.append(percentEncode("/")).append(SEPARATOR);
stringToSign.append(percentEncode(canonicalizedQueryString));

return stringToSign.toString();
}

public static String percentEncode(String value) {
try {
return value == null ? null : URLEncoder.encode(value, CHARSET_UTF8)
.replace("+", "%20").replace("*", "%2A").replace("%7E", "~");
} catch (Exception e) {
}
return "";
}

public static byte[] hmacSHA1Signature(String secret, String baseString)
throws Exception {
if (StringUtils.isEmpty(secret)) {
throw new IOException("secret can not be empty");
}
if (StringUtils.isEmpty(baseString)) {
return null;
}
Mac mac = Mac.getInstance("HmacSHA1");
SecretKeySpec keySpec = new SecretKeySpec(secret.getBytes(CHARSET_UTF8), ALGORITHM);
mac.init(keySpec);
return mac.doFinal(baseString.getBytes(CHARSET_UTF8));
}

public static String newStringByBase64(byte[] bytes)
throws UnsupportedEncodingException {
if (bytes == null || bytes.length == 0) {
return null;
}

return new String(Base64.encodeBase64(bytes, false), CHARSET_UTF8);
```

```
}

public static void main(String[] args) {
String str =
"GET&%2F&AccessKeyId%3DCdwKFNmXeHJuMOrT&Action%3DDescribeInstances&Format%3DJSON&RegionId%3
Dcn-hangzhou&SignatureMethod%3DHMAC-SHA1&SignatureNonce%3D9fdf20f2-9a32-4872-bcd4-
c6036082ebef&SignatureVersion%3D1.0&Timestamp%3D2015-12-21T09%253A05%253A44Z&Version%3D2014-
05-26";
byte[] signBytes;
try {
signBytes = SignatureUtils.hmacSHA1Signature("byczfpx4PKBzUNjjL4261cE3s6HQmH" + "&", str.toString());
String signature = SignatureUtils.newStringByBase64(signBytes);

} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}
}
```

Query Examples of Metric Data for ECS Instances

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URISyntaxException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;
import java.util.SimpleTimeZone;
import java.util.UUID;
import net.sf.json.JSON;
import net.sf.json.JSONArray;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpException;
import org.apache.commons.httpclient.HttpMethod;
import org.apache.commons.httpclient.MultiThreadedHttpConnectionManager;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.methods.PostMethod;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
public class EcsTemplate {
private final static String SIGNATURE_VERSION = "1.0";
private final static String defaultSignatureType = "HMAC-SHA1";
private final static String API_Format = "JSON";
//cms version
private final static String API_VERSION = "2015-10-20";
//Enter your AccessKey information.
private final static String accessKeyId = "YouraccessKeyId";
private final static String accessKeySecret = "YouraccessKeySecret";
private final static String domainnew = "metrics.aliyuncs.com";
protected String domain="metrics.aliyuncs.com";
//protected String domain = "alert.aliyuncs.com";
```

```
private static String formatISO8601Date(Date date) {
SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'",
Locale.US);
df.setTimeZone(new SimpleTimeZone(0, "GMT"));
return df.format(date);
}
protected HttpHeaders buildHttpHeaders(String sessionId) {
HttpHeaders headers = new HttpHeaders();
headers.setContentType(MediaType.APPLICATION_JSON);
headers.set("Authentication", sessionId);
return headers;
}
public static void main(String[] args) {
Map<String, String> parameters = new HashMap<String, String>();
//String action = "DescribeInstances";
String action = "QueryMetricList";
parameters.put("RegionId", "cn");
parameters.put("Action", action);
parameters.put("Project", "acs_ecs");
parameters.put("Metric", "CPUUtilization");
parameters.put("StartTime", "2016-01-01 00:00:00");
parameters.put("Period", "60");
parameters.put("Length", "1000");
//Note the dimension format! !
parameters.put("Dimensions", "{\"instanceId\":\"********\"}");
parameters.put("AccessKeyId", accessKeyId);
parameters.put("Format", API_Format);
parameters.put("SignatureMethod", defaultSignatureType);
parameters.put("SignatureNonce", UUID.randomUUID().toString());
parameters.put("SignatureVersion", SIGNATURE_VERSION);
parameters.put("Version",API_VERSION );
parameters.put("Timestamp", formatISO8601Date(new Date()));
String url = "http://"+domainnew;
if(!url.endsWith("/")){
url += "/";
}
url += "?";
url += UrlUtil.generateQueryString(parameters, true);
String signature = null;
try {
signature = SignatureUtils.generate("GET", parameters, accessKeySecret);
} catch (Exception e) {
e.printStackTrace();
}
url += "&Signature=" + signature;
HttpMethod http1 = new PostMethod();
HttpMethod httpMethod = new GetMethod(url);
System.out.println(url);
HttpClient httpClient = new HttpClient(new MultiThreadedHttpConnectionManager());
httpClient.getHttpConnectionManager().getParams().setConnectionTimeout(6000);//Set the request timeout time to
6 seconds.
httpClient.getHttpConnectionManager().getParams().setSoTimeout(30000); //Set the read timeout time.
try {
int statusCode = httpClient.executeMethod(httpMethod);
//if(statusCode!=200)return null;
byte[] bytes = httpMethod.getResponseBody();
```

```
String result = "[" + statusCode + "]"+new String(bytes, "UTF-8");
System.out.println(result);
} catch (HttpException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}
}
```

The following describes the examples of Java signature algorithm and C++ signature algorithm for reference.

Java signature algorithm example

Signature code:UrlUtil.java

```
import java.net.URLEncoder;
import java.util.Map;

import org.apache.commons.lang.StringUtils;
import org.apache.log4j.Logger;

public class UrlUtil {
private static Logger logger = Logger.getLogger(UrlUtil.class);

private final static String CHARSET_UTF8 = "utf8";

/**
*
* @param url
* @return
*/
public static String urlEncode(String url) {
if (!StringUtils.isEmpty(url)) {
try {
url = URLEncoder.encode(url, "UTF-8");
} catch (Exception e) {
logger.warn("Url encode error:" + e.getMessage());
}
}

return url;
}

public static String generateQueryString(Map<String, String> params, boolean isEncodeKV) {
StringBuilder canonicalizedQueryString = new StringBuilder();
for (Map.Entry<String, String> entry : params.entrySet()) {
if (isEncodeKV)
canonicalizedQueryString.append(percentEncode(entry.getKey())).append("=")
```

```
.append(percentEncode(entry.getValue())).append("&");
else
canonicalizedQueryString.append(entry.getKey()).append("=")
.append(entry.getValue()).append("&");
}
if (canonicalizedQueryString.length() > 1) {
canonicalizedQueryString.setLength(canonicalizedQueryString.length() - 1);
}
return canonicalizedQueryString.toString();
}

public static String percentEncode(String value) {
try {
// After using URLEncoder.encode for encoding, replace "+", "*", and "%7E" with values that conform to the
encoding standard specified by the API.
return value == null ? null : URLEncoder.encode(value, CHARSET_UTF8)
.replace("+", "%20").replace("*", "%2A").replace("%7E", "~");
} catch (Exception e) {
//Impossible exception
}
return "";
}
}
```

SignatureUtils.java

```


import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.Map;
import java.util.TreeMap;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;
import org.apache.commons.lang.StringUtils;


public class SignatureUtils {

private final static String CHARSET_UTF8 = "utf8";
private final static String ALGORITHM = "UTF-8";
private final static String SEPARATOR = "&";

public static Map<String, String> splitQueryString(String url)
throws URISyntaxException, UnsupportedEncodingException {
URI uri = new URI(url);
String query = uri.getQuery();
final String[] pairs = query.split("&");
TreeMap<String, String> queryMap = new TreeMap<String, String>();
for (String pair : pairs) {
```

```
final int idx = pair.indexOf("=");
final String key = idx > 0 ? pair.substring(0, idx) : pair;
if (!queryMap.containsKey(key)) {
queryMap.put(key, URLDecoder.decode(pair.substring(idx + 1), CHARSET_UTF8));
}
}
return queryMap;
}

public static String generate(String method, Map<String, String> parameter,
String accessKeySecret) throws Exception {
String signString = generateSignString(method, parameter);
System.out.println("signString---"+signString);
byte[] signBytes = hmacSHA1Signature(accessKeySecret + "&", signString);
String signature = newStringByBase64(signBytes);
System.out.println("signature---"+signature);
if ("POST".equals(method))
return signature;
return URLEncoder.encode(signature, "UTF-8");

}

public static String generateSignString(String httpMethod, Map<String, String> parameter)
throws IOException {
TreeMap<String, String> sortParameter = new TreeMap<String, String>();
sortParameter.putAll(parameter);

String canonicalizedQueryString = UrlUtil.generateQueryString(sortParameter, true);
if (null == httpMethod) {
throw new RuntimeException("httpMethod can not be empty");
}

StringBuilder stringToSign = new StringBuilder();
stringToSign.append(httpMethod).append(SEPARATOR);
stringToSign.append(percentEncode("/")).append(SEPARATOR);
stringToSign.append(percentEncode(canonicalizedQueryString));

return stringToSign.toString();
}

public static String percentEncode(String value) {
try {
return value == null ? null : URLEncoder.encode(value, CHARSET_UTF8)
.replace("+", "%20").replace("*", "%2A").replace("%7E", "~");
} catch (Exception e) {
}
return "";
}

public static byte[] hmacSHA1Signature(String secret, String baseString)
throws Exception {
if (StringUtils.isEmpty(secret)) {
throw new IOException("secret can not be empty");
}
if (StringUtils.isEmpty(baseString)) {
return null;
```

```
}
Mac mac = Mac.getInstance("HmacSHA1");
SecretKeySpec keySpec = new SecretKeySpec(secret.getBytes(CHARSET_UTF8), ALGORITHM);
mac.init(keySpec);
return mac.doFinal(baseString.getBytes(CHARSET_UTF8));
}

public static String newStringByBase64(byte[] bytes)
throws UnsupportedEncodingException {
if (bytes == null || bytes.length == 0) {
return null;
}

return new String(Base64.encodeBase64(bytes, false), CHARSET_UTF8);
}

public static void main(String[] args) {
String str =
"GET&%2F&AccessKeyId%3DCdwKFNmXeHJuMOrT&Action%3DDescribeInstances&Format%3DJSON&RegionId%3
Dcn-hangzhou&SignatureMethod%3DHMAC-SHA1&SignatureNonce%3D9fdf20f2-9a32-4872-bcd4-
c6036082ebef&SignatureVersion%3D1.0&Timestamp%3D2015-12-21T09%253A05%253A44Z&Version%3D2014-
05-26";
byte[] signBytes;
try {
signBytes = SignatureUtils.hmacSHA1Signature("byczfpx4PKBzUNjjL4261cE3s6HQmH" + "&", str.toString());
String signature = SignatureUtils.newStringByBase64(signBytes);

} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}
}
```

C++ signature algorithm example

```
#include "ali_rpc_request.h"
#include "ali_string_utils.h"
#include "ali_encode_utils.h"
#include "ali_urlencode.h"
#include "ali_log.h"
#include <stdio.h>
#include <time.h>
static std::string get_utc_string() {
time_t now;
struct tm *timenow;
now = time(&now);
timenow = gmtime(&now);
std::string res= get_format_string("%d-%02d-%02dT%02d:%02d:%02dZ", timenow->tm_year + 1900,
timenow->tm_mon + 1,
timenow->tm_mday,
timenow->tm_hour,
```

```
timenow->tm_min,
timenow->tm_sec);
return res;
}

AliRpcRequest::AliRpcRequest(std::string version,
std::string appid,
std::string secret,
std::string url)
: AliHttpRequest(url),
version_(version),
appid_(appid),
secret_(secret),
url_(url) {

}

std::string AliRpcRequest::GetSignUrl() {
std::string encoded;
std::map<std::string, std::string> mapWithPublicArgs;

time_t now;
time(&now);
this->sign_nounce = get_format_string("%ld", now);
this->utc_time_ = get_utc_string();
if(this->query_.size()) {
std::vector<std::string> vec_queries;
strsplit(this->query_, vec_queries, "&");
for(int i = 0; i < vec_queries.size(); i++) {
std::string& item = vec_queries[i];
int pos = item.find("=");
mapWithPublicArgs[item.substr(0, pos)] = item.substr(pos + 1, item.size() - pos - 1);
}
}
mapWithPublicArgs["Format"] = "JSON";
mapWithPublicArgs["Version"] = this->version_;
mapWithPublicArgs["AccessKeyId"] = this->appid_;
mapWithPublicArgs["SignatureMethod"] = "HMAC-SHA1";
mapWithPublicArgs["TimeStamp"] = utc_time_;
mapWithPublicArgs["SignatureVersion"] = "1.0";
mapWithPublicArgs["SignatureNonce"] = sign_nounce;//

for(std::map<std::string, std::string>::iterator it = mapWithPublicArgs.begin();
it != mapWithPublicArgs.end(); it++) {
if(!encoded.empty()) {
encoded.append("&");
}
append_format_string(encoded, "%s=%s", urlencode(it->first).c_str(), urlencode(it->second).c_str());
}

encoded = this->method_ + "&" + urlencode("/") + "&" + urlencode(encoded);
ALI_LOG("sign str=%s\n", encoded.c_str());
return encoded;
}

int AliRpcRequest::CommitRequest() {
```

```
std::string sign_url = GetSignUrl();
std::string sign = encode_compute_hmac_sha1(this->secret_ + "&",
(char*)sign_url.c_str(),
sign_url.size());
AddRequestQuery("TimeStamp", utc_time_);
AddRequestQuery("Format", "JSON");
AddRequestQuery("Version", this->version_);
AddRequestQuery("AccessKeyId", this->appid_);
AddRequestQuery("SignatureMethod", "HMAC-SHA1");
AddRequestQuery("SignatureVersion", "1.0");
AddRequestQuery("SignatureNonce", this->sign_nounce);
AddRequestQuery("Signature", sign);
return AliHttpRequest::CommitRequest();
}
```

Query the metric data of a specific product instance during a specified time period.

Request type

GET

Request parameters

Name	Type	Required	Description
Action	String	Yes	Operation interface name, required parameter. Value: QueryMetricList.
Project	String	Yes	Name space. It indicates the product to which the metric data belongs, for example, “acs_ecs_dashboard” or “acs_rds_dashboard” .
Metric	String	Yes	Metric name.
Period	String	No	Time interval, calculated in seconds, for example, 60, 300, or 900. If this field is not specified, raw data is queried based on the report period stated during metric registration. If this field is specified,

			corresponding statistics are queried based on the specified period.
StartTime	String	No	Start time, which can be a millisecond value counted from 00:00:00, January 1, 1970, or formatted data, for example, 2015-10-20 00:00:00.
EndTime	String	No	End time, which can be a millisecond value counted from 00:00:00, January 1, 1970, or formatted data, for example, 2015-10-20 00:00:00.
Dimensions	String	No	Key-value set used to filter metric data. The key can use one or more dimensionKeys stated during metric registration and the value relates to this key. instanceId is required. The Map object must be a JSON string. Dimensions must be input in order.
Express	String	No	The express parameter is used to calculate metric data and includes the extend keyword. This parameter is typically used to convert the unit of metric data through the four arithmetic operations. For example, in { “sum” :” sum *1000” }, “sum” is the name of the extend field and “sum* 1000” is an arithmetic operation for the extended field. Multiple extend fields are

			supported, for example, { “sumExtend” :” sum * 1000” ,” avgExtend ” :” avg / 1000” }.
Length	String	No	Size of each query, used in paging query. Default value: 1000.
Cursor	String	No	Cursor.

- For more information about how to assign values to Project, Metric, Period, and Dimensions of various cloud products, see Preset metric reference.

Parameter description

Batch query of instance metric data: To query the metric data of multiple instances, input the instance IDs in JSON format in the Dimensions parameter. Example: [{ “instanceId” :” ” }, { “instanceId” :” ” }, { “instanceId” :” ” }, { “instanceId” :” ” }]

For more information about how to assign values to Project, Metric, Period and Dimensions, see Preset metric reference.

startTime and endTime are in left-open and right-closed mode, and startTime cannot be the same as or later than endTime.

Cursor is a parameter in paging mode. The Cursor parameter indicates there is a next page. If the return result is null, no more pages available.

Generally, the Period value can be 60 seconds (1 minute), 300 seconds (5 minutes), or 900 seconds (15 minutes). You can set the Period parameter based on the document requirements and query scenario. For example, to query data for a specified day, if you set Period to 60, then 1,000 data items are returned. (Actually, 1,440 items are available but only 1,000 items are returned since the maximum return value cannot be greater than 1,000.) If you set Period to 300, then 288 data items are returned.

The interface supports RAM subaccount calls. The operation descriptor during authorization is “cms:QueryMetricList” and the resource descriptor is “*” .

Return parameters

Name	Type	Description

Period	String	Time interval, always calculated in seconds, for example, 60, 300, or 900.
Cursor	String	Cursor.
Datapoints	String	Metric data list, in the following format: { “timestamp”: 1490164200000, “Maximum”: 100, “userId”: “1234567898765432”, “Minimum”: 4.55, “instanceId”: “i-bp18abl200xk9599ck7c”, “Average”: 93.84 }.
Code	String	Status code. Code 200 is returned if no exception occurs.
Success	Boolean	Whether the current query is successful. If there is an exception on the server side, the returned value is “false” (and “true” otherwise).
Message	String	Status description. The message is null when Code is 200.
RequestId	String	If any issue occurs with querying, you can provide this field to a technical engineer to troubleshoot.

Error code

Error code	Description	Meaning
400	Bad Request	Syntactic error of the client request
403	Forbidden	Not authorized
404	Not Found	Client error, not found
500	Internal Server Error	Server error
200	OK	Normal

Examples

- Request example

```
http://metrics.cn-hangzhou.aliyuncs.com/?Action=QueryMetricList
&EndTime=2017-05-17+11%3A30%3A27
&StartTime=2017-05-17+11%3A20%3A27
&Period=60
&Dimensions=%7B%22instanceId%22%3A%22i-abcdefgh123456%22%7D
&Timestamp=2017-03-22T09%3A30%3A57Z
&Project=acs_ecs_dashboard
&Metric=cpu_idle
&express={"extend":{"avgExtend":"Average*10"}}
```

- Return example

XML format

```
<QueryMetricListResponse>
<Period>60</Period>
<Datapoints>
<Datapoints>
<timestamp>1490152860000</timestamp>
<Maximum>100</Maximum>
<userId> 1234567898765432</userId>
<Minimum>93.1</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.52</Average>
<avgExtend>995.2</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490152920000</timestamp>
<Maximum>100</Maximum>
<userId> 1234567898765432 </userId>
<Minimum>92.59</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.49</Average>
<avgExtend>994.9</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490152980000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>92.86</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.44</Average>
<avgExtend>994.4</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490153040000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>91.43</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.36</Average>
<avgExtend>993.6</avgExtend>
</Datapoints>
```

```
<Datapoints>
<timestamp>1490153100000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>93.55</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.51</Average>
<avgExtend>995.1</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490153160000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>93.1</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.52</Average>
<avgExtend>995.2</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490153220000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>92.59</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.42</Average>
<avgExtend>994.2</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490153280000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>91.18</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.34</Average>
<avgExtend>993.4</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490153340000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>92.86</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.46</Average>
<avgExtend>994.6</avgExtend>
</Datapoints>
<Datapoints>
<timestamp>1490153400000</timestamp>
<Maximum>100</Maximum>
<userId>1234567898765432</userId>
<Minimum>91.18</Minimum>
<instanceId>i-abcdefgh123456</instanceId>
<Average>99.35</Average>
<avgExtend>993.5</avgExtend>
</Datapoints>
</Datapoints>
<RequestId>6661EC50-8625-4161-B349-E0DD59002AB7</RequestId>
```

```
<Success>true</Success>
<Code>200</Code>
</QueryMetricListResponse>
```

JSON format

```
{
"Period": "60",
"Datapoints": [
{
"timestamp": 1490152860000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 93.1,
"instanceId": "i-abcdefgh123456",
"Average": 99.52
"avgExtend": 995.2
},
{
"timestamp": 1490152920000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 92.59,
"instanceId": "i-abcdefgh123456",
"Average": 99.49
"avgExtend": 994.9
},
{
"timestamp": 1490152980000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 92.86,
"instanceId": "i-abcdefgh123456",
"Average": 99.44
"avgExtend": 994.4
},
{
"timestamp": 1490153040000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 91.43,
"instanceId": "i-abcdefgh123456",
"Average": 99.36
"avgExtend": 993.6
},
{
"timestamp": 1490153100000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 93.55,
"instanceId": "i-abcdefgh123456",
"Average": 99.51
"avgExtend": 995.1
},
{
```

"timestamp": 1490153160000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 93.1,
"instanceId": "i-abcdefgh123456",
"Average": 99.52
"avgExtend": 995.2
},
{
"timestamp": 1490153220000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 92.59,
"instanceId": "i-abcdefgh123456",
"Average": 99.42
"avgExtend": 994.2
},
{
"timestamp": 1490153280000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 91.18,
"instanceId": "i-abcdefgh123456",
"Average": 99.34
"avgExtend": 993.4
},
{
"timestamp": 1490153340000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 92.86,
"instanceId": "i-abcdefgh123456",
"Average": 99.46
"avgExtend": 994.6
},
{
"timestamp": 1490153400000,
"Maximum": 100,
"userId": "1234567898765432",
"Minimum": 91.18,
"instanceId": "i-abcdefgh123456",
"Average": 99.35
"avgExtend": 993.5
}
],
"RequestId": "6A5F022D-AC7C-460E-94AE-B9E75083D027",
"Success": true,
"Code": "200"
}

Description

Call interfaces to install the CloudMonitor agent on a specified ECS instance. For details about the

agent, refer to Agent overview.

NOTE

- Before using interfaces, ensure that the Server Guard agent has been installed on your instance. The Server Guard agent is currently integrated into security images. It is installed by default on purchased ECS instances. You can log on to the Server Guard Console to check that the agent is running on each server.
- The success rate of installing the CloudMonitor agent via interfaces is about 95%. If installation fails, log on to the instance to install the agent manually.

Request method

GET

Request parameters

Name	Type	Required?	Description
Action	String	Yes	Required parameter; value: NodeInstall
UserId	String	Yes	ID of an Alibaba Cloud user
InstanceId	String	Yes	Instance ID, for example, i-22jja5c2l
Force	Boolean	No	Whether to install the CloudMonitor agent forcibly. If the agent has been installed, the default value indicates forced installation.

Return parameters

Name	Type	Description
Success	Boolean	Whether the request is successful
ErrorCode	Integer	Request failure status code. The value 200 indicates that the request is successful, and a non-200 value indicates that the request fails.
RequestId	String	Requested UUID, which is used for log query

ErrorMessage	String	Request failure prompt message

Error code

Error code	Description	Meaning
400	Bad Request	Syntactic error of the client request
403	Forbidden	Not authorized
404	Not Found	Client error, not found
500	Internal Server Error	Server error
200	OK	Normal

Examples

- Request example

```
http://metrics.cn-hangzhou.aliyuncs.com/?Action=NodeInstall
&InstanceId= i-abcdefgh123456
&Force=true
&UserId= 1234567898765432
&<Public request parameters>
```

- Return example

XML format

```
<NodeInstallResponse>
<ErrorCode>200</ErrorCode>
<Success>true</Success>
</NodeInstallResponse>
```

JSON format

```
{
"ErrorCode": 200,
"Success": true
}
```

Note:

“Sum” is the sum of values obtained in a statistical period. Take ECS Internet traffic measurement

as an example. The unit is KB/min and the statistical period is 5 minutes. Then the returned result is the total traffic in 5 minutes. Interfaces of the new version support query of ECS basic metric data, whereas interfaces of the old version do not. OpenAPI supports the query of metric data from the last 31 days. Dimension for each parameter is a JSON string, for example, {"instanceId":"i-23gyb3kkd"}.

ECS metric reference

Basic metrics

No agent is required for obtaining and querying ECS basic metric data.

The project is named “acs_ecs_dashboard”, the sampling period is 60s, and the Period value is 60 or an integer multiple of 60.

The instanceId value for Dimensions is the instanceId of the ECS instance.

Metric	Description	Unit	Dimensions	Statistics
CPUUtilization	CPU usage	Percent	instanceId	Average, Minimum, and Maximum
InternetInRate	Inbound public network bandwidth	bits/s	instanceId	Average, Minimum, and Maximum
IntranetInRate	Inbound private network bandwidth	bits/s	instanceId	Average, Minimum, Maximum
InternetOutRate	Outbound public network bandwidth	bits/s	instanceId	Average, Minimum, and Maximum
IntranetOutRate	Outbound private network bandwidth	bits/s	instanceId	Average, Minimum, and Maximum
InternetOutRate_Percent	Outbound public network bandwidth usage	%	instanceId	Average
DiskReadBPS	Total system disk read BPS	Bps	instanceId	Average, Minimum, and Maximum
DiskWriteBPS	Total system disk write BPS	Bps	instanceId	Average, Minimum, and Maximum
DiskReadIOPS	System disk	Count/Second	instanceId	Average,

	read IOPS			Minimum, and Maximum
DiskWriteIOPS	System disk write IOPS	Count/Second	instanceId	Average, Minimum, and Maximum
VPC_PublicIP_InternetInRate	VPC - Inbound public network bandwidth	bits/s	instanceId	Average, Minimum, and Maximum
VPC_PublicIP_InternetOutRate	VPC - Outbound public network bandwidth	bits/s	instanceId	Average, Minimum, and Maximum
VPC_PublicIP_InternetOutRate_Percent	VPC - Outbound public network bandwidth usage	%	instanceId	Average

Operating system metrics

Metrics of the new agent version

- The minimum period value is 15s.

Metric	Description	Unit	Dimensions	Statistics
cpu_idle	Host.cpu.idle, percentage of CPU in the idle state	%	instanceId	Average, Minimum, and Maximum
cpu_system	Host.cpu.system, CPU usage of the current kernel space	%	instanceId	Average, Minimum, and Maximum
cpu_user	Host.cpu.user, CPU usage of the current user space	%	instanceId	Average, Minimum, and Maximum
cpu_wait	Host.cpu.iowait, percentage of CPU waiting for I/O operation	%	instanceId	Average, Minimum, and Maximum
cpu_other	Host.cpu.other, CPU usage of other items. Other types of CPU consumption are calculated using the	%	instanceId	Average, Minimum, and Maximum

	formula Nice + SoftIrq + Irq + Stolen.			
cpu_total	Host.cpu.total, current total CPU usage	%	instanceId	Average, Minimum, and Maximum
memory_totalspace	Host.mem.total, total memory	bytes	instanceId	Average, Minimum, and Maximum
memory_usedspace	Host.mem.used, memory in use. The calculation formula is as follows: Memory occupied by user programs + Buffers + Cached. Buffers indicates the memory occupied by the buffer, and cached indicates the memory occupied by the cache.	bytes	instanceId	Average, Minimum, and Maximum
memory_actualusedspace	Host.mem.actualused, memory occupied by the user. The calculation formula is as follows: Used – Buffers – Cached.	bytes	instanceId	Average, Minimum, and Maximum
memory_freespace	Host.mem.free, available memory	bytes	instanceId	Average, Minimum, and Maximum
memory_freeutilization	Host.mem.free utilization, percentage of available memory	%	instanceId	Average, Minimum, and Maximum
memory_usedutilization	Host.mem.used utilization, memory usage	%	instanceId	Average, Minimum, and Maximum
load_1m	Host.load1, average system	None	instanceId	Average, Minimum, and

	load during the past minute. This metric is not available in Windows.			Maximum
load_5m	Host.load5, average system load during the past 5 minutes. This metric is not available in Windows.	None	instanceId	Average, Minimum, and Maximum
load_15m	Host.load15, average system load during the past 15 minutes. This metric is not available in Windows.	None	instanceId	Average, Minimum, and Maximum
diskusage_used	Host.diskusage.used, disk space in use	bytes	instanceId, device	Average, Minimum, and Maximum
diskusage_utilization	Host.disk.utilization, disk usage	%	instanceId, device	Average, Minimum, and Maximum
diskusage_free	Host.diskusage.free, available disk space	bytes/s	instanceId, device	Average, Minimum, and Maximum
diskusage_total	Host.diskussage.total, total disk storage	bytes	instanceId, device	Average, Minimum, and Maximum
disk_readbytes	Host.disk.readbytes, number of bytes read from the disk per second	bytes/s	instanceId, device	Average, Minimum, and Maximum
disk_writebytes	Host.disk.writebytes, number of bytes written to the disk per second	bytes/s	instanceId, device	Average, Minimum, and Maximum
disk_readiops	Host.disk.readiops, number of read requests sent to the disk per second	Requests/s	instanceId, device	Average, Minimum, and Maximum
disk_writeiops	Host.disk.writeiops, number of write requests sent to the disk	Requests/s	instanceId, device	Average, Minimum, and Maximum

	per second			
fs_inodeutilization	Host.fs.inode, inode usage	%	instanceId, device	Average, Minimum, and Maximum
networkin_rate	Host.netin.rate, number of bits received by the network adapter per second, that is, the uplink bandwidth of the network adapter	bits/s	instanceId, device	Average, Minimum, and Maximum
networkout_rate	Host.netout.rate, number of bits sent by the network adapter per second, that is, the downlink bandwidth of the network adapter	bits/s	instanceId, device	Average, Minimum, and Maximum
networkin_packages	Host.netin.packages, number of packets received by the network adapter per second	Packets/s	instanceId, device	Average, Minimum, and Maximum
networkout_packages	Host.netout.packages, number of packets sent by the network adapter per second	Packets/s	instanceId, device	Average, Minimum, and Maximum
networkin_errorpackages	Host.netin.errorpackage, number of incoming error packets detected by the drive	Packets/s	instanceId, device	Average, Minimum, and Maximum
networkout_errorpackages	Host.netout.errorpackages, number of outgoing error packets detected by the drive	Packets/s	instanceId, device	Average, Minimum, and Maximum
net_tcpconnecti	Host.tcpconnec	Connections	instanceId,	Average,

on	tion, number of TCP connections in various states, including LISTEN, SYN_SENT, ESTABLISHED, SYN_RECV, FIN_WAIT1, CLOSE_WAIT, FIN_WAIT2, LAST_ACK, TIME_WAIT, CLOSING, and CLOSED		state	Minimum, and Maximum

ApsaraDB for RDS metric reference

The project is named “acs_rds_dashboard”, the default sampling period is 300s, and the Period value is 300 or an integer multiple of 300.

The minimum sampling period is 60s if 1-minute monitoring frequency is enabled on the RDS Console.

The instanceId value for Dimensions is the instanceId of the RDS instance.

Metric	Description	Unit	Dimensions	Statistics
CpuUsage	CPU usage	Percent	instanceId, type= CpuUsage	Average, Minimum, and Maximum
DiskUsage	Disk usage	Percent	instanceId, type= DiskUsage	Average, Minimum, and Maximum
IOPSUsage	IOPS usage	Percent	instanceId, type= IOPSUsage	Average, Minimum, and Maximum
ConnectionUsage	Connection usage	Percent	instanceId, type= ConnectionUsage	Average, Minimum, and Maximum
DataDelay	Read-only instance latency	Seconds	instanceId, type=DataDelay	Average, Minimum, and Maximum
MemoryUsage	Memory usage	Percent	instanceId, type=	Average, Minimum, and

			MemoryUsage	Maximum
MySQL_NetworkInNew	MysqlInbound network traffic per second	bits/s	instanceId	Average, Minimum, and Maximum
MySQL_NetworkOutNew	MysqlOutbound network traffic per second	bits/s	instanceId	Average, Minimum, and Maximum
SQLServer_NetworkInNew	SQLServer - inbound network traffic per second	bits/s	instanceId	Average, Minimum, and Maximum
SQLServer_NetworkOutNew	SQLServer - outbound network traffic per second	bits/s	instanceId	Average, Minimum, and Maximum

Server Load Balancer metric reference

The project is named “acs_slb_dashboard”, the sampling period is 60s, and the Period value is 60 or an integer multiple of 60.

The instanceId value for Dimensions is the instanceId of the Server Load Balancer instance.

The port value for Dimensions is the Server Load Balancer instance port number.

The vip value for Dimensions is the Server Load Balancer instance service address.

Layer-4 protocol Metrics

Metric	Description	Unit	Dimensions	Statistics
HeathyServerCount	Number of healthy backend ECS instances	Count	instanceId	Average, Minimum, and Maximum
UnhealthyServerCount	Number of faulty backend ECS instances	Count	instanceId	Average, Minimum, and Maximum
PacketTX	Number of outgoing packets per second on the port	Count/Second	instanceId, port, vip	Average, Minimum, and Maximum

PacketRX	Number of incoming packets per second on the port	Count/second	instanceId, port, vip	Average, Minimum, and Maximum
TrafficRXNew	Inbound data volume per second on the port	bits/s	instanceId, port, vip	Average, Minimum, and Maximum
TrafficTXNew	Outbound data volume per second on the port	bits/s	instanceId, port, vip	Average, Minimum, and Maximum
ActiveConnection	Number of active connections on the port, that is, the number of connections that clients set up to access Server Load Balancer	Count	instanceId, port, vip	Average, Minimum, and Maximum
InactiveConnection	Number of inactive connections on the port, that is, the number of connections that are idle after access to Server Load Balancer	Count	instanceId, port, vip	Average, Minimum, and Maximum
NewConnection	Current number of new connections on the port	Count	instanceId, port, vip	Average, Minimum, and Maximum
MaxConnection	Number of concurrent connections on the port	Count	instanceId, port, vip	Average, Minimum, and Maximum
DropConnection	Number of dropped connections per second during monitoring	Count/Second	instanceId, port, vip	Average, Minimum, and Maximum
DropPacketRX	Number of dropped incoming packets per second during	Count/Second	instanceId, port, vip	Average, Minimum, and Maximum

	monitoring			
DropPacketTX	Number of dropped outgoing packets per second during monitoring	Count/Second	instanceId, port, vip	Average, Minimum, and Maximum
DropTrafficRX	Number of dropped incoming bits per second during monitoring	bits/s	instanceId, port, vip	Average, Minimum, and Maximum
DropTrafficTX	Number of dropped outgoing bits per second during monitoring	bits/s	instanceId, port, vip	Average, Minimum, and Maximum
InstanceActiveConnection	Number of active connections per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceDropConnection	Number of dropped connections per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceDropPacketRX	Number of dropped incoming packets per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceDropPacketTX	Number of dropped outgoing packets per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceDropTrafficRX	Number of dropped incoming bits per second on the instance	bits/s	instanceId	Average, Minimum, and Maximum
InstanceDropTrafficTX	Number of dropped outgoing bits per second on the instance	bits/s	instanceId	Average, Minimum, and Maximum

InstanceInactiveConnection	Number of inactive connections per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceMaxConnection	Maximum number of concurrent connections per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceNewConnection	Number of new connections per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstancePacketRX	Number of incoming packets per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstancePacketTX	Number of outgoing packets per second on the instance	Count/Second	instanceId	Average, Minimum, and Maximum
InstanceTrafficRX	Number of incoming bits per second on the instance	bits/s	instanceId	Average, Minimum, and Maximum
InstanceTrafficTX	Number of outgoing bits per second on the instance	bits/s	instanceId	Average, Minimum, and Maximum

Layer-7 protocol Metrics

Metric	Description	Unit	Dimensions	Statistics
Qps	Layer-7 protocol port QPS	Count/Second	instanceId, port, vip	Average
Rt	Layer-7 protocol port RT	ms	instanceId, port, vip	Average
StatusCode2xx	Layer-7 protocol port status code 2XX	Count/Second	instanceId, port, vip	Average
StatusCode3xx	Layer-7 protocol port	Count/Second	instanceId, port, vip	Average

	status code 3XX			
StatusCode4xx	Layer-7 protocol port status code 4XX	Count/Second	instanceId, port, vip	Average
StatusCode5xx	Layer-7 protocol port status code 5XX	Count/Second	instanceId, port, vip	Average
StatusCodeOther	Layer-7 protocol port status code others	Count/Second	instanceId, port, vip	Average
UpstreamCode4xx	Layer-7 protocol port Upstream status code 4xx	Count/Second	instanceId, port, vip	Average
UpstreamCode5xx	Layer-7 protocol port Upstream status code 5xx	Count/Second	instanceId, port, vip	Average
UpstreamRt	Layer-7 protocol port UpstreamRT	ms	instanceId, port, vip	Average
InstanceQps	Layer-7 protocol instance QPS	Count/Second	instanceId	Average
InstanceRt	Layer-7 protocol instance RT	ms	instanceId	Average
InstanceStatusCode2xx	Layer-7 protocol instance status code 2XX	Count/Second	instanceId	Average
InstanceStatusCode3xx	Layer-7 protocol instance status code 3XX	Count/Second	instanceId	Average
InstanceStatusCode4xx	Layer-7 protocol instance status code 4XX	Count/Second	instanceId	Average
InstanceStatusCode5xx	Layer-7 protocol instance status code 5XX	Count/Second	instanceId	Average
InstanceStatus	Layer-7	Count/Second	instanceId	Average

CodeOther	protocol instance status code Other			
InstanceUpstreamCode4xx	Layer-7 protocol instance Upstream status code 4XX	Count/Second	instanceId	Average
InstanceUpstreamCode5xx	Layer-7 protocol instance Upstream status code 5XX	Count/Second	instanceId	Average
InstanceUpstreamRt	Layer-7 protocol instance UpstreamRT	ms	instanceId	Average

OSS metric reference

For details, refer to OSS metric reference.

EIP metric reference

The project is named “acs_vpc_eip”, the sampling period is 60s, and the Period value is 60 or an integer multiple of 60.

The instanceId value for Dimensions is the instanceId of the EIP instance.

The ip value for Dimensions is the EIP instance IP address.

Metric	Description	Unit	Dimensions	Statistics
net_tx.rate	Outbound bandwidth	Byte/s	instanceId	
net_rx.rate	Inbound bandwidth	Byte/s	instanceId	
net.txPkgs	Number of outgoing packets	Count	instanceId	Average, Minimum, and Maximum
net.rxPkgs	Number of incoming packets	Count	instanceId	Average, Minimum, and Maximum

ApsaraDB for Redis metric reference

The project is named “acs_kvstore”, the sampling period is 60s, and the Period value is 60 or an integer multiple of 60.

The instanceId value for Dimensions is the instanceId of the ApsaraDB for Redis instance.

Metric	Description	Unit	Dimensions	Statistics
MemoryUsage	Percentage of memory in use	Percent	instanceId	Average, Minimum, and Maximum
ConnectionUsage	Percentage of connections in use	percent	instanceId	Average, Minimum, and Maximum
IntranetInRatio	Percentage of bandwidth consumed during write operations	percent	instanceId	Average, Minimum, and Maximum
IntranetOutRatio	Percentage of bandwidth consumed during read operations	percent	instanceId	Average, Minimum, and Maximum
IntranetIn	Write speed	bits/s	instanceId	Average, Minimum, and Maximum
IntranetOut	Read speed	bits/s	instanceId	Average, Minimum, and Maximum
FailedCount	Number of failed operations on ApsaraDB for Redis	Count/second	instanceId	Average, Minimum, and Maximum
CpuUsage	CPU usage	percent	instanceId	Average, Minimum, and Maximum
UsedMemory	Memory in use	Bytes	instanceId	Average, Minimum, and Maximum

Message Service metric reference

The project is named “acs_messageservice_new”, the sampling period is 300s, and the Period value is 300 or an integer multiple of 300.

The region value for Dimensions is the region where the queue is located.

The queue value for Dimensions is the name of the queue.

Metric	Description	Unit	Dimensions	Statistics
ActiveMessages	Number of active messages	Count	region,queue	Average, Minimum, and Maximum
InactiveMessages	Number of inactive messages	Count	region,queue	Average, Minimum, and Maximum
DelayMessages	Number of delayed messages	Count	region,queue	Average, Minimum, and Maximum

Alibaba Cloud CDN metric reference

The project is named “acs_cdn”, the sampling period is 300s, and the Period value is 300 or an integer multiple of 300.

The instanceId value for Dimensions is the CDN domain.

Metric	Description	Unit	Dimensions	Statistics
QPS	Queries per second, that is, total access requests in a specific time interval/Time interval	Count	instanceId	Average, Minimum, and Maximum
BPS	Peak bandwidth, that is, maximum network traffic per unit time	bits/s	instanceId	Average, Minimum, and Maximum
hitRate	Bytes hit rate, that is, the	Percent	instanceId	Average, Minimum, and

	probability that request bytes hit the cache in a specific time interval			Maximum
code4xx	Percentage of HTTP Return Code 4xx in a specific time interval	Percent	instanceId	Average, Minimum, and Maximum
code5xx	Percentage of HTTP Return Code 5xx in a specific time interval	Percent	instanceId	Average, Minimum, and Maximum
InternetOut	Downstream traffic	Bytes	Average, Minimum, and Maximum	

ApsaraDB for MongoDB metric reference

The project is named “acs_mongodb”, the sampling period is 300s, and the Period value is 300 or an integer multiple of 300.

The role value for Dimensions is “Primary” or “Secondary”, which indicates the master or slave instance.

Metric	Metric name	Description	Unit	Dimensions	Statistics
CPUUtilization	CPU usage	CPU usage of the instance	%	userId, instanceId, role	Average, Minimum, and Maximum
MemoryUtilization	Memory usage	Memory usage of the instance	%	userId, instanceId, role	Average, Minimum, and Maximum
DiskUtilization	Disk usage	Disk usage of the instance	%	userId, instanceId, role	Average, Minimum, and Maximum
IOPSUtilization	IOPS usage	IOPS usage of the instance	%	userId, instanceId, role	Average, Minimum, and Maximum
Connection	Connection	Percentage	%	userId,	Average,

Utilization	usage	of connections in use		instanceId, role	Minimum, and Maximum
QPS	Average SQL queries per second	Average number of SQL queries per second on the MongoDB instance	Queries	userId, instanceId, role	Average, Minimum, and Maximum
Connection Amount	Connections in use	Current number of connections that applications have established with the MongoDB instance	Connections	userId, instanceId, role	Average, Minimum, and Maximum
InstanceDiskAmount	Disk space used by instance	Disk space used by the instance itself	Bytes	userId, instanceId, role	Average, Minimum, and Maximum
DataDiskAmount	Disk space used by data	Disk space used by data	Bytes	userId, instanceId, role	Average, Minimum, and Maximum
LogDiskAmount	Disk space used by logs	Disk space used by logs	Bytes	userId, instanceId, role	Average, Minimum, and Maximum
IntranetIn	Inbound network traffic	Inbound network traffic of the instance	Bytes	userId, instanceId, role	Average, Minimum, and Maximum
IntranetOut	Outbound network traffic	Outbound network traffic of the instance	Bytes	userId, instanceId, role	Average, Minimum, and Maximum
NumberRequests	Request count	Total number of requests sent to the server	Requests	userId, instanceId, role	Average, Minimum, and Maximum
OpInsert	Insert operation count	Total number of insert commands received since the	Times	userId, instanceId, role	Average, Minimum, and Maximum

		last time MongoDB was started			
OpQuery	Query operation count	Total number of query commands received since the last time MongoDB was started	Times	userId, instanceId, role	Average, Minimum, and Maximum
OpUpdate	Update operation count	Total number of update commands received since the last time MongoDB was started	Times	userId, instanceId, role	Average, Minimum, and Maximum
OpDelete	Delete operation count	Total number of delete commands executed since the last time MongoDB was started	Times	userId, instanceId, role	Average, Minimum, and Maximum
OpGetmore	Getmore operation count	Total number of getmore commands executed since the last time MongoDB was started	Times	userId, instanceId, role	Average, Minimum, and Maximum
OpCommand	Command operation count	Total number of commands sent to the database since the last time MongoDB was started	Times	userId, instanceId, role	Average, Minimum, and Maximum

ExpressConnect metric reference

The project is named “acs_express_connect “, the sampling period is 60s, and the Period value is 60 or an integer multiple of 60.

The instanceId value for Dimensions is the interface ID of the ExpressConnect router.

Metric	Metric name	Unit	Dimensions
IntranetRX	Inbound network traffic	Bytes	instanceId
IntranetTX	Outbound network traffic	Bytes	instanceId
ReceiveBandwidth	Inbound network bandwidth	bit/s	instanceId
TransportedBandwidth	Outbound network bandwidth	bit/s	instanceId