
# Hitachi HiCommand® Backup Services Manager

## SDK
## Developers Guide

Aug 19th, 2005

Version *5.0.03*

Doc ID: MK-95APT006-01

# Table of Contents

**INTRODUCTION** ..... **3**

**HITACHI HICOMMAND® BACKUP SERVICES MANAGER SDK** ..... **4**

SOAP Web Services ..... 4
Obtaining WSDL ..... 4
Basic HBSM Web Service Structure ..... 5
Authentication ..... 5
Session Management ..... 6

**EXAMPLE CLIENTS** ..... **7**

PERL Client ..... 7
JAVA Client ..... 9

**PUBLISHED WEB SERVICES** ..... **11**

BackupSummary ..... 11

# Introduction

This document provides developers with an overview of the SDK functionality within the Hitachi HiCommand® Backup Services Manager Software.

If you have a question about the SDK, please contact the Hitachi Data Systems Technical Support Center who will be happy to assist you:

Technical Support Center Contact Information

| | |
|---|---|
| Phone: | |
| Nth & Latin America | 1-800-348-4357 |
| Europe | +(44)-175-361-8000 |
| Asia Pacific | +(61)-2-9325-3300 |
| Email: | support@hds.com |

# Hitachi HiCommand® Backup Services Manager SDK

The HBSM SDK (Software Developers Kit) is implemented as a set of Web Services (WS). These Web Services communicate via HTTP and SOAP. SOAP allows objects to talk to each other in a distributed, decentralized, Web-based environment.

## SOAP Web Services

A SOAP client is a program that creates an XML document containing the information needed to invoke remotely a method in a distributed system. SOAP clients need not be traditional. In addition to being a desktop application, a SOAP client could also be a Web server or a server-based application.

Messages and requests from SOAP clients are typically sent over HTTP. As a result, SOAP documents are able to traverse almost any firewall, enabling the exchange of information across divergent platforms.

A SOAP server is simply special code that listens for SOAP messages and acts as a distributor and interpreter of SOAP documents. External Web services may interact with application servers, which process SOAP requests from a variety of clients.

SOAP servers ensure that documents received over a HTTP connection are converted to a language that the object at the other end understands. Because all communications are made in the form of XML, objects in one language (say, Java) may communicate through SOAP with objects in any other language (PERL, for example). It's the job of the SOAP server to make sure the end points understand, and are happy with, the SOAP they're being served.

## Obtaining WSDL

All of the HBSM Web Services publish WSDL (Web Services Description Language) describing the parameters passed and returned. WSDL for any service can be obtained by appending ?WSDL to the end of the SOAP end point.

For security purposes you will be required to authenticate before WSDL will be returned.

## Basic HBSM Web Service Structure

Each HBSM Web Service is structured in a similar manner and returns a similar set of data. Each WS has two main methods:

- The getXXX() method.
- The getNext() method.

A call to an HBSM WS will take the form of a call to the main getXXX() method for the web service. An example of this is the call to getBackupSummary() for the BackupSummary WS.

The call to the method will return a multi dimensional array containing records of data. The first element in the array are the records, and the second the data. If more data is present on the server than can be sensibly returned in one SOAP message then the client will require to make subsequent calls to the getNext() method to retrieve the remainder of the data.

The client should continue making getNext() calls until the last element of the returned array is null.

## Authentication

The HBSM WS are authenticated via HTTP Basic authentication in the HTTP header. The account and password details should be passed as a colon separated Base64 encoded string in the Authorization header. Following is an example of a well formed HTTP header passing authentication details:

```
POST /services/BackupSummary HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime,
multipart/related, text/*
User-Agent: Axis/1.1
Host: localhost:8090
SOAPAction: ""
Content-Length: 1094
Authorization: Basic U09BUHlTdGV2ZTpYXNzd29yZA==
```

## Session Management

As a call to an HBSM WS may require multiple calls to the WS to retrieve all the data held on the server the client must maintain state with the server. This is achieved by the passing of cookies. The client should return any cookies passed back to it by the server.

Following is an example of a conversation between the server and client showing cookie passing to maintain state.

**Client to Server**

```
POST /services/BackupSummary HTTP/1.0
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime,
multipart/related, text/*
User-Agent: Axis/1.1
Host: localhost:8090
SOAPAction: ""
Content-Length: 1094
Authorization: Basic U09BUHlTdGV2ZTpYXNzd29yZA==
```

**Server to Client**

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Connection: close
Date: Fri, 21 May 2004 03:43:26 GMT
Server: Apache Tomcat/4.0.6 (HTTP/1.1 Connector)
Set-Cookie:
JSESSIONID=A74A74D7EB78FFDDAE88BCF866C0B059;Path=/
```

**Client to Server**

```
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, application/dime,
multipart/related, text/*
User-Agent: Axis/1.1
Host: localhost:8090
SOAPAction: ""
Content-Length: 346
Authorization: Basic U09BUHlTdGV2ZTpwYXNzd29yZA==
Cookie: JSESSIONID=A74A74D7EB78FFDDAE88BCF866C0B059
```

# Example Clients

Following are example calls to the BackupSummary WS with PERL & Java.

## PERL Client

The following client example makes a call to the BackupSummary WS using the SOAP::Lite Perl libraries. Authentication is achieved by passing the account and password on the url to the end point. Sessions are managed by the use of HTTP cookies.

```
#!perl -w
#
# Perl packages required for this sample program:
# SOAP::Lite from  http://soaplite.com/download.html
#       Other dependant packages from http://search.cpan.org/:
#               Crypt::SSLeay             for HTTPS/SSL
#               LWP::UserAgent, URI       for SOAP::Transport::HTTP::Client
#               HTTP::Cookies
#               SOAP::DateTime
#               Date::Manip

# require http cookies for WS session support
use HTTP::Cookies;

use SOAP::Lite
    xmlschema => 2001;

# required for easy manipulation of SOAP date time
use SOAP::DateTime;

# set timezone for ConvertDate function
Date::Manip::Date_Init("TZ=PST");

my ($soap, $response);

# make connection to the soap service, when authenticating use the username:password
# convention on the URL
# Note: If the user id contains an @ character it will need to be escaped. e.g. admin@aptare
# would be passed as 'http://admin%40aptare.com:password@host/services/BackupSummary'
$soap = SOAP::Lite
        -> uri('http://localhost:80/services/BackupSummary')
        -> proxy('http://user:password@host/services/BackupSummary',
               cookie_jar => HTTP::Cookies->new(ignore_discard => 1));

# construct Lists for parameters

my @clientIdList;
$clientIdList[0] = SOAP::Data->type(long => 203960);
$clientIdList[1] = SOAP::Data->type(long => 202066);

my @serverIdList;
$serverIdList[0] = SOAP::Data->type(long => 32);
$serverIdList[1] = SOAP::Data->type(long => 234);
$serverIdList[2] = SOAP::Data->type(long => 1234);

my @jobTypeList;
$jobTypeList[0] = SOAP::Data->type(long => 32);
$jobTypeList[1] = SOAP::Data->type(long => 234);
$jobTypeList[2] = SOAP::Data->type(long => 1234);

my @jobSummaryList;

$jobSummaryList[0] = SOAP::Data->type(long => 32);
$jobSummaryList[1] = SOAP::Data->type(long => 234);
```

```
$jobSummaryList[2] = SOAP::Data->type(long => 1234);

my @jobStatusList;
$jobStatusList[0] = SOAP::Data->type(long => 32);
$jobStatusList[1] = SOAP::Data->type(long => 234);
$jobStatusList[2] = SOAP::Data->type(long => 1234);

my @statusExcludeList;
$statusExcludeList[0] = SOAP::Data->type(long => 32);

# Call the SOAP endpoint
$response = $soap->getBackupSummary(
# the following two lines uses the SOAP::DateTime functions to convert a more meaningful # date
time convention to SOAP
        SOAP::Data->name(startDateTime =>
                SOAP::Data->type(dateTime => ConvertDate("12/14/2003 12:14:37"))),
        SOAP::Data->name(finishDateTime =>
                SOAP::Data->type(dateTime => ConvertDate("12/14/2004 12:14:37"))),
        # ConvertDate is from SOAP::Date package
        SOAP::Data->name(normalizedTimeZone => "PST"),
        SOAP::Data->name(groupID => SOAP::Data->type(long => 200147)),
        SOAP::Data->name(cascade => SOAP::Data->type(boolean => "1")),
        # boolean => "1" = true, "0" = false

        SOAP::Data->name(clientIDList =>
                SOAP::Data->type('SOAP-ENC:Array' => \@clientIdList)),
        SOAP::Data->name(serverIDList =>
                SOAP::Data->type('SOAP-ENC:Array' => \@serverIdList)),
        SOAP::Data->name(jobTypeList =>
                SOAP::Data->type('SOAP-ENC:Array' => \@jobTypeList)),
        SOAP::Data->name(jobSummaryList =>
                SOAP::Data->type('SOAP-ENC:Array' => \@jobSummaryList)),
        SOAP::Data->name(jobStatusList =>
                SOAP::Data->type('SOAP-ENC:Array' => \@jobStatusList)),
        SOAP::Data->name(statusExcludeList =>
                SOAP::Data->type('SOAP-ENC:Array' => \@statusExcludeList))
);

my $status = 1;
while ($status == 1) {

   if ($response->fault) {
      print "Fault: ", $response->faultdetail, "\n";
      print "String: ", $response->faultstring, "Code: ", $response->faultcode, "\n";
      $status = -1;
   }
   else {
      # we have data
      my @rows = @{$response->result};
      # @rows is the array of structs, if row is null then no more rows coming from
      # the web service

      foreach my $row (@rows) {
         if (defined($row)) {
             print "----------------------------------------\n";
             for ($j = 0; $j < @$row; $j++) {
                print $row->[$j], "\n";
             }
         }
         else { # row is null so no more rows
             $status = 0;
             print "No more rows\n";
             last;
         }
      }
      if ($status != 0) {
         # we got here so there must be another chunk of data. Get more rows
         print "Get more rows\n";
         $response = $soap->getNext();
      }
   }
```

```
}
```

## JAVA Client

The following client example makes a call to the BackupSummary WS using the Apache Axis libraries. Authentication is achieved by the setUsername() and setPassword() on the Call object. Sessions are managed by call to setMaintainSession(true).

```
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import javax.xml.rpc.ParameterMode;
import javax.xml.rpc.ServiceException;
import java.rmi.RemoteException;

import java.util.Date;
import java.net.MalformedURLException;
import java.util.Calendar;

import com.aptare.ws.*;

public class BackupSummaryClient {


  public void getBackupSummary(String endPoint) {
    Service   service;
    Call      call;
    Object[]  values = new Object[11];

    //GregorianCalendar aoStartDate = new GregorianCalendar(int year, int month, int day,
    int hour, int minute, int second); month starts from 0 - 11
    GregorianCalendar aoStartDate = new GregorianCalendar(2004, 6, 1, 12, 0);
    GregorianCalendar aoFinishDate = new GregorianCalendar(2004, 6, 9, 12, 0);
    Date startDateTime = aoStartDate.getTime();
    Date finishDateTime = aoFinishDate.getTime();
    String normalizedTimeZone = null;
    long groupID = 200147;
    boolean cascade = true;
    long[] clientIdList = {203960, 202066};
    long[] serverIdList = {203960, 202066};
    long[] jobTypeList = null;
    long[] jobSummaryList = null;
    long[] jobStatusList = null;
    long[] statusExcludeList = {0};

    try {
      service = new Service();
      call = (Call) service.createCall();
      call.setTargetEndpointAddress(new java.net.URL(endPoint));
      call.setMaintainSession(true);

      call.setOperationName("getBackupSummary");
      call.addParameter("startDateTime", XMLType.XSD_DATETIME, ParameterMode.IN);
      call.addParameter("finishDateTime", XMLType.XSD_DATETIME, ParameterMode.IN);
      call.addParameter("normalizedTimeZone", XMLType.XSD_STRING, ParameterMode.IN);
      call.addParameter("groupID", XMLType.XSD_LONG, ParameterMode.IN);
      call.addParameter("cascade", XMLType.XSD_BOOLEAN, ParameterMode.IN);
      call.addParameter("clientIDList", XMLType.XSD_ANY, ParameterMode.IN);
      call.addParameter("serverIDList", XMLType.XSD_ANY, ParameterMode.IN);
      call.addParameter("jobTypeList", XMLType.XSD_ANY, ParameterMode.IN);
      call.addParameter("jobSummaryList", XMLType.XSD_ANY, ParameterMode.IN);
      call.addParameter("jobStatusList", XMLType.XSD_ANY, ParameterMode.IN);
      call.addParameter("statusExcludeList", XMLType.XSD_ANY, ParameterMode.IN);
      call.setReturnType(XMLType.XSD_ANY);

      call.setUsername("User");
      call.setPassword("password");
```

```
      values[0] = startDateTime;
      values[1] = finishDateTime;
      values[2] = normalizedTimeZone;
      values[3] = new Long(groupID);
      values[4] = new Boolean(cascade);
      values[5] = longArrayToObject(clientIdList);
      values[6] = longArrayToObject(serverIdList);
      values[7] = longArrayToObject(jobTypeList);
      values[8] = longArrayToObject(jobSummaryList);
      values[9] = longArrayToObject(jobStatusList);
      values[10] = longArrayToObject(statusExcludeList);
      //Call 1
      System.out.println("Making SOAP call to " + endPoint + "(ListBackupSummary)");
      Object[] o = (Object[]) call.invoke(values);
      Object[] o2 = (Object[]) o[0];
      System.out.println("Call 1 returned " + o.length + " rows");

      //Call 2
      //Check that the last record was not null
      if (o[o.length-1] != null) {
         call.setOperationName("getNext");
         call.removeAllParameters();
         values = new Object[0];
         o = (Object[]) call.invoke(values);
         o2 = (Object[]) o[0];
         System.out.println("Call 2 returned " + o.length + " rows");
      }

    }
    catch (ServiceException e) {
      System.out.println("ServiceException: " + e.getMessage());
    }
    catch (MalformedURLException e) {
      System.out.println("MalformedURLException: " + e.getMessage());
    }
    catch (RemoteException e) {
      System.out.println("RemoteException: " + e.getMessage());
    }
  }

  private Long[] longArrayToObject(long[] array) {
    Long[]  objectArray;

    if (array == null) {
      return null;
    }
    objectArray = new Long[array.length];
    for (int i=0; i < array.length; i++) {
      objectArray[i] = new Long(array[i]);
    }
    return objectArray;
  }


}
```

# Published Web Services

## BackupSummary

**Web Service:** BackupSummary
**End Point:** http://hbsmportal.*yourdomain*.com/services/ListBackupSummary
**WSDL:** http:// hbsmportal.*yourdomain*.com/services/ListBackupSummary?wsdl

**Method:** getBackupSummary

**Parameters:**

| Type | Name | Description |
|---|---|---|
| date | startDateTime | Reporting period start time. See code examples for appropriate format. |
| date | finishDateTime | Reporting period finish time. Reporting period start time. See code examples for appropriate format. |
| string | normalizedTimeZone | TimeZone to use for generating the report |
| long | groupID | The Server GroupId for which to generate the report. This can be NULL if either clientIDList or serverIDList is not NULL |
| boolean | cascade | Cascade the server groups if set to true (i.e. 1) |
| long[] | clientIDList | List of Client Ids for which to generate reports. This can be NULL if either groupID or serverIDList is not NULL. |
| long[] | serverIDList | The Server ID for which to generate the report. This generates a report for all the clients associated with the Server. This can be NULL if either groupID or clientIDList is not NULL. |
| long[] | jobTypeList | NULL indicates all jobs<br>1 - Full Backup<br>2 - Incremental Backup<br>3 - Application Backup<br>4 - Catalog Backup<br>5 - Archive<br>6 - Restore<br>7 - Verify |
| long[] | jobSummaryList | NULL indicates all job summary statuses<br>0 - Ok<br>1 - Warning<br>2 - Error |
| long[] | jobStatusList | List of NetBackup Status Codes. NULL indicates all job statuses. |
| long[] | statusExcludeList | List of NetBackup Status Codes to exclude |

**Return:**

A two dimensional array of anyType. Dimension 1 contains records for each backup job. Dimension 2 contains the following:

| | Type | Name | Description |
|---|---|---|---|
| [0] | long | jobID | Record Id of job in HBSM database |
| [1] | long | nbuJobID | Record Id of job in the Backup Management system |
| [2] | int | jobType | Can be one of<br>1 - Full Backup<br>2 - Incremental Backup<br>3 - Application Backup<br>4 - Catalog Backup<br>5 - Archive<br>6 - Restore<br>7 - Verify |
| [3] | int | summaryStatus | Can be one of<br>NULL - In Process<br>0 - Ok<br>1 - Warning<br>2 - Error |
| [4] | int | jobStatus | Backup Management system Status Code. Please refer to Backup Management system documentation. |
| [5] | long | clientID | Record Id of the client server in HBSM database |
| [6] | string | clientInternalName | Internal name of client server |
| [7] | string | clientExternalName | External name of client server |
| [8] | long | serverID | Record Id of the management server in HBSM database |
| [9] | string | serverInternalName | Internal name of management server |
| [10] | string | serverExternalName | External name of management server |
| [11] | date | startTime | Start time of the job |
| [12] | date | finishTime | Finish time of the job |
| [13] | int | productType | Specifies the Backup Management system, which can be one of the following<br>1 - Veritas NetBackup |
| [14] | long | kilobytes | Size of data being backed up or restored |
| [15] | long | fileCount | Number of files backed up or restored |
| [16] | long | policyID | Record Id of the policy used to |

| | Type | Name | Description |
|---|---|---|---|
| | | | initiate the backup/restore job, stored in HBSM database |
| [17] | string | policyName | Name of policy used to initiate backup/restore job |
| [18] | long | scheduleID | Record Id of the schedule used to initiate the backup/restore job, stored in HBSM database |
| [19] | string | scheduleName | Name of schedule used to initiate backup/restore job |

**Notes:**

This method will return a maximum of 100 records per call. If the last element in Dimension 1 of the array is not null then more records reside on the server. Additional records can be retrieved by calling getNext().

**Method:** getNext

**Parameters:** none

**Return:** The same as getBackupSummary.

**Notes:**

Returns remaining records from a getBackupSummary call. If the last element in dimension 1 of the array is not null then more records reside on the server. Additional records can be retrieved by calling getNext().