

SCS³ Services

User's Guide

Copyright 2011 by The University of Alabama in Huntsville (UAH)
Information Technology and Systems Center (ITSC)
All rights reserved.

Permission to use, copy, and modify this software and its documentation for any purpose, without fee, is hereby granted, under the following conditions:

1. The source code, as a whole or part, may not be redistributed to any other individual or organization without written permission of UAH ITSC.
2. Modifications made to the source code should be reported to UAH ITSC before the organization receiving the source code distributes the intellectual property. UAH ITSC will then have the option of including the modifications within the baseline code maintained at UAH ITSC.
3. Results of the implementation of this software will be reported to UAH ITSC.

The above copyright notice, this paragraph, and the following three paragraphs must appear in all copies, modifications, and distributions.

Created by the Information Technology and Systems Center, University of Alabama in Huntsville, sponsored by NASA Earth Science Enterprise. For technical information, contact info@itsc.uah.edu

IN NO EVENT SHALL UAH BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF UAH HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

UAH SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE AND ACCOMPANYING DOCUMENTATION, IF ANY, PROVIDED HEREUNDER IS PROVIDED "AS IS". UAH HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

TABLE OF CONTENTS

Introduction	1
Interface Description	3
Satellite Services	5
Using the SCS³ Services	7
Creating a Subscription	7
Maintaining Subscriptions	7
Performing Historical Searches.....	7
The SCS³ Services	9
Login – scs3login.pl.....	9
Add a New Subscription – scs3addsubscription.pl	10
Retrieve Subscriptions – scs3getsubscriptions.pl.....	11
Modify a Subscription – scs3modsubscription.pl	12
Get Datasets for Satellites – scs3getdatasets.pl	12
Perform a Coincidence Search – scs3search.pl	13
Perform an ECHO Search – echosearch.pl	14

INTRODUCTION

The original Coincidence Search routine was written a long time ago, in a galaxy far, far away. It was part of a couplet of programs that provided not only coincidences for satellite nadirs, but plots of satellite orbits. The orbit plotter fell into disrepair, as it was based on evolving web technologies, but the Coincidence Search routine survived, eventually becoming an “engine”.

The original orbit model was provided (in a roundabout way) by the Colorado Center for Astrodynamical Research (CCAR) in Boulder Colorado. It served us well for a decade and a half. However, during the latest revision of the CSE, it was discovered that *only* the “SGP4” algorithm should be used with the NORAD-produced TLEs. The latest Coincidence Search Engine (CSE) uses the SGP4 algorithm. It produces slightly different results than the CCAR orbit model. It is not clear which is “correct”.

The Earth Observing System Clearing House (ECHO) provides users with a mechanism for searching and retrieving data granules (files) from datasets produced by a wide variety of platforms, including satellites. It was thought that it would be beneficial to end-users to combine the coincidence search with the granule search, providing a “one-step” solution to finding data for user-specified events (coincidences). From this idea, SCS³ was born.

The SCS³ implements a *subscription service*. The full-up implementation consists of a *front-end*, which provides a web-based user interface; *services* which implement the front-end code and perform historical searches; and a *server* which implements ongoing subscriptions. End-users may use the front-end to define either a one-time *historical search* or a *subscription*. The former searches for past coincidences and then searches ECHO for granules. The latter continuously searches for coincidences from a user-defined starting date through an ending date, searching ECHO for granules and reporting “hits” to the end-user via e-mail. The major distinction is that the historical search is done *once* for events in the past, whereas the subscription is run *repeatedly* for events in the future.

The front-end interface is documented online (<http://scs3.nsstc.nasa.gov>) and is pretty much self-explanatory. The server process runs continuously, monitoring for new coincidences for subscriptions; it is not user-accessible. This document provides instructions for programmatic use of the SCS³ services which implement subscription submission and historical searches.

Note the distinction between the *services* which are accessible by URLs and the *server*, which is not.

INTERFACE DESCRIPTION

All of the services described herein use the REST interface. That is, the *request* is transmitted using the HTTP GET protocol¹ which specifies a *function* to be performed and a list of *name – value* pairs that provide the function parameters.

The root URL is as follows:

```
https://ghrc.nsstc.nasa.gov/services/scs3/function[?name=value[&name=value]...]
```

function indicates the operation to be performed, and the *name=value* parameters modify that function. For example,

```
https://ghrc.nsstc.nasa.gov/services/scs3/scs3login.pl?oldornew=old
&useremail=john.doe%40mycompany.com&userpass=xxxxxxx
```

This is a login request (scs3login.pl) for an existing user (oldornew=old) identified as “useremail=john.doe@mycompany.com” with a password of “userpass=xxxxxxx”. Note that there are *no* quote marks surrounding the values.

Most special characters must be translated into escape sequences. This is done by prefixing the ASCII value (in hexadecimal) with a percent sign (%). In the example above, the “@” was translated to “%40”. Typically, any special characters other than period (.), hyphen (-), or underscore (_) must be escaped. However, the punctuation required by the GET interface protocol – namely the first question mark (?), equals signs (=) between the parameter and the value, and the ampersands (&) which separate the parameters – must be specified exactly as shown, not escaped.

The response to any request is most often XML. “Most often” because some errors may cause the request to be rejected before it gets to the service function in which case HTML or plain text may be returned. It is suggested that the user inspect the first few characters of the response to determine the response type or to check the MIME type returned, if possible. If XML is returned, the response string should begin with “<?xml version=...”.

Continuing the example above, the response might be

```
<?xml version="1.0" encoding="UTF-8"?>
<SCS3LoginResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
    "http://ghrc.nsstc.nasa.gov/html/services/scs3/SCS3Services.xsd"
  result="success"
  userid="AAAAjjjkkkk111MMMMNNN"/>
```

In this case, the user login was accepted (result="success") and the user-id was returned.

In subsequent chapters, the XML header (“<?xml...”) and namespace information (“xmlns:...” and “xsi:noNameSpaceSchemaLocation...”) are not shown for brevity. User-written message parsers should ignore

¹ If a command string is overlong (greater than 10K bytes), POST protocol may also be used.

these entities. The line breaks and indentations shown in the examples may or may not be present in the responses.

SATELLITE SERVICES

REST services have been written to access the satellite information. Although these services are not actually part of the SCS³ interface, they may be needed to translate satellite *names* to *ids*. Most of the SCS³ services require numeric satellite ids.

The services be accessed using REST (GET protocol). For example,

```
https://ghrc.nsstc.nasa.gov/services/satellites/satellites.pl?satid=25063
```

This fetches the information for satellite 25063 (TRMM). To get information about *all* satellites, omit the “satid”.

The “satellites.pl” service provides information about satellites that we “track” (i.e., have TLEs for). The available GET parameters are as follows:

<code>satid=<i>id</i></code>	The satellite-id (optional)
<code>satname=<i>name</i></code>	The satellite name (optional)
<code>inecho={yes no}</code>	“yes” to only select satellites with ECHO data
<code>l0data={yes no}</code>	“yes” to only select satellites with level-0 datasets
<code>l1data={yes no}</code>	“yes” to only select satellites with level-1 datasets
<code>l2data={yes no}</code>	“yes” to only select satellites with level-2 datasets
<code>l3data={yes no}</code>	“yes” to only select satellites with level-3 datasets
<code>l4data={yes no}</code>	“yes” to only select satellites with “other” datasets

The response is one of the following:

```
<Satellites>
  <satellite
    id="integer"
    name="string"
    from="datetime"
    [thru="datetime"]
    [period="float"]
    altitude="float"
    [information="string"]
    [comments="string"]
    inecho="{Y|N}"
    counts="int,int,int,int,int,int"/>...
</Satellites>
```

A successful search will yield one or more <satellite> entries. The “id” and “name” attributes provide a mapping between the satellite-id and the satellite name.

The “from” and optional “thru” attributes indicate the date range of *data* from the science instruments, which is not necessarily the launch to de-orbit range. Dates are supplied as *yyyy-mm-ddThh:mm:ssZ*. The “thru” attribute is returned only for decommissioned satellites.

The “period” attribute is the nominal orbital period in minutes. It will not be returned for geosynchronous satellites, like GOES. The “altitude” attribute is the nominal altitude of the satellite above sea level in kilometers.

The “information” string, if present, contains a URL to the satellite’s web page, and the “comments” string, if present, contains a locally-produced comment about the satellite.

The “inecho” attribute has been added to support SCS³. If set to “Y”, it indicates that ECHO has datasets corresponding to this satellite. The “counts” attribute lists the number of datasets in ECHO for processing level 0, 1, 2, 3, 4, and “other” (unspecified).

A failure response looks like:

```
<Satellites>
  <error msg="string"/>
</Satellites>
```

The contents of the “msg” attribute in the <error> tag will indicate the nature of the error.

For example,

```
https://ghrc.nsstc.nasa.gov/services/satellites/satellites.pl?satid=25063
```

Returns

```
<Satellites>
  <satellite
    id="25063"
    name="TRMM"
    from="1997-11-29"
    period="92.391"
    altitude="397.75"
    information="http://trmm.gsfc.nasa.gov/"
    inecho="Y"
    counts="0,2,15,31,7,23"/>
</Satellites>
```

Similarly,

```
https://ghrc.nsstc.nasa.gov/services/satellites/satellites.pl?satname=TRMM
```

returns the same result.

To quickly build a listing of the satellites that may be used in SCS³ queries, use

```
https://ghrc.nsstc.nasa.gov/services/satellites/satellites.pl?inecho=yes
```

This query, if successful, will return a <Satellites> tag containing <satellite> tags for all of the satellites for which there are data in ECHO.

USING THE SCS³ SERVICES

The SCS3 services, described in the following chapter, allow the user to create and maintain subscriptions and search for coincidences and data. The latter is called a “historical search”, since the search must be in the past. By definition, subscriptions must be in the future.

CREATING A SUBSCRIPTION

To create a subscription, the following steps must be performed:

- Create an account *or* login using an existing account (scs3login.pl). The login process will return a “userid” string which must be used in subsequent service calls.
- Determine which *satellites* are to be polled for coincidences. The satellite list may be *a priori* knowledge, or else the satellites.pl service described in the previous chapter may be used to determine the satellite names and ids.
- Determine which *datasets* are to be polled for data granules. This can be done using the scs3getdatasets.pl service.
- Gather the rest of the information needed for the subscription:
 - The unique *subscription name*, a descriptive title of up to 80 characters.
 - The *starting date and time* for the subscription.
 - The *ending date and time* for the subscription.
 - The *frequency* at which to run the subscription.
 - The length of time to wait for new data (*latency*).
 - The *area-of-interest* for satellite/ground coincidences (optional).
 - The *proximity range* for satellite/satellite coincidences (optional).
- Use the scs3addsubscription.pl service to create the subscription.

Thereafter, e-mail will be sent whenever a coincidence is found for which there are data granules.

MAINTAINING SUBSCRIPTIONS

Once logged on using scs3login.pl, existing users may get a list of their subscriptions using the scs3getsubscriptions.pl service and may update them using the scs3modsubscription.pl service.

PERFORMING HISTORICAL SEARCHES

To perform a historical search, perform the following steps:

- Determine which *satellites* are to be polled for coincidences. The satellite list may be *a priori* knowledge, or else the satellites.pl service described in the previous chapter may be used to determine the satellite names and ids.
- Determine which *datasets* are to be polled for data granules. This can be done using the scs3getdatasets.pl service.

- Gather the rest of the information needed for the search:
 - The *starting date and time* for the search.
 - The *ending date and time* for the search.
 - The *area-of-interest* for satellite/ground coincidences (optional).
 - The *proximity range* for satellite/satellite coincidences (optional).
- Use the `scs3search.pl` service to perform the historical search.

The `echosearch.pl` service can also be used to poll ECHO for granules for coincidences; however, this is unnecessary, as the `scs3search.pl` script does this automatically.

THE SCS³ SERVICES

The SCS³ services provide the REST service access for the Satellite Coincidence Search Subscription Service. It is invoked by the SCS³ front end and can be used by external processes that do not wish to use the SCS³ front end. Services are invoked using GET protocol thusly:

```
https://ghrc.nsstc.nasa.gov/services/scs3/service?options
```

Note that the services are on the secure port since we are passing around password information.

The following services are defined:

LOGIN – SCS3LOGIN.PL

This module provides login services for SCS³. New or returning users must “login” to access the other services. This is done using the following options:

<code>oldornew={old new}</code>	“old” for a returning user, “new” for a new user
<code>userid=string</code>	The user’s rowid (see below for this use)
<code>useremail=string</code>	The user’s e-mail address which serves as the name
<code>userpass=string</code>	The user’s password
<code>userfirst=string</code>	The user’s first name (optional)
<code>userlast=string</code>	The user’s last name (optional)
<code>userorg=string</code>	The user’s organization (optional)

A new user will use the “oldornew=new”, “useremail”, and “userpass” options. The password string must be 8-12 characters and must contain at least one upper-case letter, one lower-case letter, one number, and one special character. The “userfirst”, “userlast”, and “userorg” options are not required, but desired.

A returning user will use “oldornew=old” and at least “useremail”. If no “userpass” value is present, the service will use the “useremail” address to send the user’s password. Otherwise, the “userpass” value is verified with the information in the database.

The “userid” option is only used by the front end code to quickly verify the identity of a user as he/she moves between the web pages. This allows the front end to pass a single, longish string between screens instead of the user’s e-mail address and password.

If successful, the return from the REST service is XML formatted thusly:

```
<SCS3LoginResponse
  result="success"
  userid="string"/>
```

The “userid” value must be used in calls to the other SCS³ services.

If unsuccessful, the return from the REST service is

```
<SCS3LoginResponse
  result="failure"
  reason="string"/>
```

The “reason” value indicates the reason for the error.

An example:

```
https://ghrc.nsstc.nasa.gov/services/scs3/scs3login.pl?oldornew=old&
useremail=johndoe%40aol.com&userpass=1Q%40w3E%24r
```

which returns

```
<?xml version="1.0" encoding="UTF-8"?>
<SCS3LoginResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ghrc.nsstc.nasa.gov/html/services
  /scs3login.xsd"
  result="success" userid="AAAjrTAAFAAAABDjAAA"/>
```

Note that the special characters in the query must be escaped prior to transmission.

ADD A NEW SUBSCRIPTION – SCS3ADDSUBSCRIPTION.PL

This routine establishes a new subscription for a logged-in user. Guest users may not create subscriptions. The following options are available:

<code>userid=string</code>	The user-id returned by the login process
<code>subname=string</code>	The user-supplied subscription name
<code>fromdate=yyyy-mm-ddT hh:mm:ssZ</code>	The starting date of the subscription, GMT
<code>thru date=yyyy-mm-ddT hh:mm:ssZ</code>	The ending date of the subscription, GMT
<code>frequency=period</code>	The frequency at which to run the subscription
<code>latency=period</code>	How long to wait for data
<code>interval=hh:mm:ss</code>	The poll interval for the subscription (optional)
<code>aoi=float,float,float,float</code>	The area of interest for the search (optional)
<code>within=integer</code>	The proximity of satellite nadirs for searches (optional)
<code>satid1=integer</code>	The first satellite id
<code>satname1=string</code>	The first satellite name
<code>dsname1=string</code>	The first dataset name
<code>dsosddurl1=string</code>	The first dataset OSDD URL
<code>satid2=integer</code>	The second satellite id (optional)
<code>satname2=string</code>	The second satellite name (optional)
<code>dsname2=string</code>	The second dataset name (optional)
<code>dsosddurl2=string</code>	The second dataset OSDD URL (optional)
...	

The “userid” must be the value that was return by the “login” service, described above.

The “subname” is the user-specified name for the subscription. A user’s subscription names must be unique, but may duplicate those created by others.

The “fromdate” and “thru date” indicate the period of performance for the subscription. These options are required.

The “frequency” and “latency” values are specified as ISO period strings; e.g., “Pnu”, where “n” is an integer and “u” is the units. We only support a small number of unit values, namely “H” (hours), “D” (days), “W” (weeks), and “M” (months). “frequency” specifies how often the subscription is to be run between the “fromdate” and “thru date” values. “latency” specifies how long after the detection of an event to poll ECHO for data. The latter is needed because ECHO rarely has data for a very recent event.

The “interval”, “aoi”, and “within” values are passed to the CSE. If unspecified, “interval” defaults to ten seconds (“00:00:10”), “aoi” defaults to the entire globe (“-180,-90,180,90”), and “within” defaults to zero (unspecified). The “interval” values must be four floating-point numbers separated by commas, but no spaces. The order of coordinates is “west,south,east,north”. The origin (0,0) is the equator at Greenwich, with values increasing northward and eastward. This means that the valid range for longitudes is -180 to 180 and the valid range for latitudes is -90 to 90. The “north” value must exceed the “south” value. If the “west” value exceeds the “east” value, then the area-of-interest spans the date line.

The rest of the parameters are an (often long) set of tuples: satellite id, satellite name, dataset (long) name, and dataset OSDD URL. At least one tuple must be specified for each subscription.

Once again, note that any special characters in the options must be escaped before transmission. This is particularly important for the dataset OSDD URL strings.

A successful response looks like:

```
<SCS3AddSubscriptionResponse result="success" subid="string"/>
```

The “subid” value is the subscription-id which may be used in later service calls.

If the service failed, the response looks like:

```
<SCS3AddSubscriptionResponse result="failure" reason="string"/>
```

The “reason” string indicates the reason for failure.

RETRIEVE SUBSCRIPTIONS – SCS3GETSUBSCRIPTIONS.PL

This service is used to retrieve one or more of a user’s subscriptions. The options are as follows:

<code>userid=string</code>	The user-id returned by the login process
<code>subid=string</code>	An optional subscription-id string

A successful response looks like the following:

```
<SCS3GetSubscriptionsResponse result="success">
  [<Subscription subid="string" subname="string" entered="datetime"
    fromdate="datetime" thru date="datetime" frequency="string"
    latency="string" aoi="float,float,float,float" within="integer"
    interval="time" lastrun="datetime" enabled="string"/>]...
</SCS3GetSubscriptionsResponse>
```

Zero or more <Subscription> tags may be returned. All of the parameters shown are returned, even if they were not specified at the time the subscription was created. “lastrun” indicates the last time the subscription was run by the SCS³ server; it will be “1901-01-01T00:00:00Z” if the subscription has never run. “enabled” is either “Y” (yes), “N” (no), or “X” (canceled).

If the “subid” parameter was specified, the result will contain only one <Subscription> tag.

A failure response looks like:

```
<SCS3GetSubscriptionsResponse result="failure" reason="string">
</SCS3GetSubscriptionsResponse>
```

The “reason” string indicates the reason for failure.

MODIFY A SUBSCRIPTION – SCS3MODSUBSCRIPTION.PL

This service is used to change certain parameters of a subscription. The options are as follows:

<code>userid=string</code>	The user-id returned by the login process
<code>subid=string</code>	The subscription-id string
<code>toggle=on</code>	If the subscription is to be enabled/disabled (optional)
<code>delete=on</code>	If the subscription is to be (logically) deleted (optional)
<code>fromdate=yyyy-mm-ddT hh:mm:ssZ</code>	The starting date of the subscription (optional)
<code>thru date=yyyy-mm-ddT hh:mm:ssZ</code>	The ending date of the subscription (optional)
<code>frequency=period</code>	The polling frequency for the subscription (optional)
<code>latency=period</code>	How long to continue polling ECHO for data (optional)
<code>aoi=west,south,east,north</code>	The bounding box for searches (optional)
<code>within=integer</code>	The proximity of satellite nadirs in kilometers (optional)
<code>dsname1=string</code>	The first dataset to keep
<code>dsname2=string</code>	The second dataset to keep (optional)
...	

The “userid” and “subid” identify the user and the subscription. They are required.

“toggle” and “delete” are mutually exclusive. If “toggle” is specified and the subscription is disabled (“N”), it will be enabled (“Y”), and vice-versa. If “delete” is specified, the subscription will be logically deleted (“X”).

The “fromdate” and “thru date” values control the period of performance of the subscription. Changing the “fromdate” of an active subscription is rather pointless.

“frequency” and “latency” are described in the “add subscription” section.

“aoi” and “within” can be specified to alter the search parameters for the CSE.

The remainder of the options are an (often long) list of dataset OSDD URLs. There must be one entry for each dataset to *keep* in the subscription. Any other datasets will be removed from the subscription list.

There is no way to modify the “satid” or “satname” values in a subscription, nor is there any way to *add* datasets to a subscription. To do that, submit another subscription.

A successful response looks like:

```
<SCS3UpdateSubscriptionResponse result="success"/>
```

A failure response looks like:

```
<SCS3UpdateSubscriptionResponse result="failure" reason="string"/>
```

The contents of the “reason” attribute indicate the reason for failure.

GET DATASETS FOR SATELLITES – SCS3GETDATASETS.PL

This service can be used to get a list of datasets in ECHO corresponding to search criteria. The parameters are as follows:

<code>satid=integer</code>	The satellite-id
<code>satname=string</code>	The satellite name
<code>from=yyyy-mm-ddT hh:mm:ssZ</code>	The starting date/time of the search
<code>thru=yyyy-mm-ddT hh:mm:ssZ</code>	The ending date/time of the search
<code>level1=on</code>	If <i>only</i> L1 datasets are to be returned

level2=on	If <i>only</i> L2 datasets are to be returned
level3=on	If <i>only</i> L3 datasets are to be returned
level5=on	If <i>only</i> “other” datasets are to be returned
aoi= <i>west,south,east,north</i>	The bounding box for searches (optional)

One of “satid” or “satname” must be specified as well as the “from” and “thru” parameters.

The “level n ” parameters may be specified to restrict the result set to only level-1, level-2, level-3, and/or “other” (level5) datasets. Any combination may be specified. If omitted, levels 1-3 will be returned.

The area-of-interest may be specified to restrict the search to datasets covering that bounding box.

A successful response looks like:

```
<SCS3DatasetsResponse>
  [<Dataset>
    <name>string</name>
    <level>string</level>
    <osddurl>string</osddurl>
    [<documentation>string</documentation>]
    [<comment>string</comment>]
  </Dataset>]...
</SCS3DatasetsResponse>
```

Zero or more <Dataset> tags may be returned. Each will indicate the dataset name, level, and its OSDD URL. The latter is required to perform a coincidence using the scs3search.pl service, described below.

A failure response looks like:

```
<SCS3DatasetsResponse>
  <error msg="string"/>
</SCS3DatasetsResponse>
```

The contents of the “msg” attribute indicate the reason for failure.

PERFORM A COINCIDENCE SEARCH – SCS3SEARCH.PL

This service is used to search for coincidences and then retrieve ECHO data for them. The parameters are as follows:

subid= <i>string</i>	The subscription-id string (optional)
from= <i>yyyy-mm-ddT hh:mm:ssZ</i>	The starting date/time of the search
thru= <i>yyyy-mm-ddT hh:mm:ssZ</i>	The ending date/time of the search
interval= <i>hh:mm:ss</i>	The date/time polling increment
satids= <i>integer[,integer]...</i>	Satellite ids of 1-4 satellites
aoi= <i>west,south,east,north</i>	The bounding box for searches (optional)
within= <i>integer</i>	The proximity of satellite nadirs in kilometers (optional)
osddurls= <i>string[,string]...</i>	A list of dataset OSDD URLs

If “subid” is specified, it must refer to an existing subscription. This parameter is used by the SCS³ front-end to do one-time historical searches and by the SCS³ server to process subscriptions.

The “from” and “thru” parameters are required. They bracket the date and time of the search.

aoi=west,south,east,north
osddurls=string[,string]...
hits=date,date[,date,date]...

The bounding box for searches (optional)
A list of dataset OSDD URLs
A list of “hit” date *pairs*

If “subid” is specified, it must refer to an existing subscription. This parameter is used by the SCS³ server to process subscriptions.

“aoi” specifies the optional bounding box for searches (optional). The value must consist of four floating-point values, separated by commas, but no spaces. The order of the values is “west,south,east,north”. The west and east (longitude) values must range from -180 (dateline) eastward to 180 (dateline). The north and south (latitude) values must range from -90 (south pole) to 90 (north pole). The south value must not exceed the north value. If the west value exceeds the east value, then the area-of-interest spans the dateline.

“osddurls” consists of a list of *dataset-level* OSDD URLs for ECHO datasets, separated by commas. These values can be obtained by using the ECHO OpenSearch interface to do a dataset-level search. That search will return the dataset-level OSDDs for each dataset.

Since the “osddurls” parameter may contain all manner of characters that are invalid in a URL, it must be escaped using the conventional HTML escapes. (E.g., “/” becomes “%2F”.)

“hits” must contain a list of date/time *pairs* in the format *yyyy-mm-ddThh:mm:ssZ,yyyy-mm-ddThh:mm:ssZ*. These specify the date/time range to use when searching in ECHO.

A successful response looks like:

```
<ECHOSearchResponse>
  [<Coincidence from="datetime" thru="datetime">
    <Dataset name="string">
      [<file url="string" size="float"/>]...
    </Dataset>...
  </Coincidence>]...
</ECHOSearchResponse>
```

Zero or more <Coincidence> tags may be returned, indicating the date/time span of a coincidence. Within each tag may be one or more <Dataset> tags, indicating that ECHO data were found for the given coincidence for the indicated dataset. Within each <Dataset> tag may be found zero or more <file> tags which give the URL and size of the granule in MiB.

Note that a successful response may be empty, indicating no coincidences. It may also contain coincidences, but no file URLs if no data were found for the dataset in ECHO.

A failure response looks like:

```
<ECHOSearchResponse>
  <error msg="string"/>
</ECHOSearchResponse>
```

The contents of the “msg” attribute indicate the reason for failure.