



METACARTA APPLIANCE

INGESTION GUIDE

Last modified on June 8, 2007

Version 3.8

Copyright 2001 - 2007 MetaCarta, Incorporated. All rights reserved. Pat. 7,117,199. This product or document contains the confidential and proprietary information of MetaCarta, Incorporated ("MetaCarta"). This product or document is protected by copyright and distributed under license restricting its use, copying, distribution, disclosure and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of MetaCarta and its licensors, if any. Third-party software is copyrighted and licensed from MetaCarta suppliers.

Copyrighted and licensed third-party software used in this product includes: Debian Linux, which in turn includes the Open Source projects Tomcat Application Server and Perl and Python languages; software developed by Apache Software Foundation (<http://www.apache.org/>); DOM4J project from Metastuff Ltd.; and the Boost Regex Library from Dr. John Hammond (<http://www.boost.org>).

MetaCarta, the MetaCarta logo, CartaTrees, geOdrive, GeoTagger, GeoParser, and MetaCarta GTS are proprietary trademarks of MetaCarta that may be protected by one or more registrations. Other trademarks and service marks are the property of their respective owners.

All computer software provided by MetaCarta to the Government is Commercial Computer Software, as defined in Section 12.212 of the Federal Acquisition Regulation (48 CFR 12.212 (October 1995)) and Sections 227.7202-1 and 227.7202-3 of the Defense Federal Acquisition Regulation Supplement (48 CFR 227.7202-1, 227.7202-3 (June 1995)). Any Technical Data provided by MetaCarta shall be considered a "Commercial item" as defined in the FAR. The Government shall accept the standard commercial license for the commercial software and shall take no more than limited rights in any Technical Data describing any commercial item, component or process. In the event that, for any reason, the foregoing are deemed not applicable, the Government's right to use, duplicate or disclose any software shall be limited to "Restricted Rights" as defined in 48 CFR Section 52.227-19(c)(1) and (2) (June 1987), or DFARS 252.227-7014(a)(14) (June 1995), as applicable. Manufacturer is MetaCarta, Inc., 350 Massachusetts Avenue, 4th Floor, Cambridge, MA 02139

ALL PRODUCTS OR DOCUMENTATION ARE PROVIDED "AS IS" AND ALL EXPRESSED OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMER IS HELD TO BE LEGALLY INVALID.

About MetaCarta

MetaCarta, a pioneering geographic information application solutions provider, offers comprehensive analysis tools to enable better business decisions. Founded by a team of MIT researchers in 1999, MetaCarta provides groundbreaking technology that bridges the gap between Geographic Information Systems (GIS) and text search. The company presents innovative solutions that improve decision-making by making it possible for customers to discover, visualize, and act on important information. The company is privately held, with offices in Cambridge MA, Vienna VA, and Houston TX. To learn more about MetaCarta, please visit www.metacarta.com.

MetaCarta, Inc.
350 Massachusetts Avenue, 4th Floor
Cambridge MA 02139

Phone: + 1 617 661 6382
Fax: + 1 617 661 6384
Email: info@metacarta.com

Overview

The MetaCarta GTS appliance indexes documents and allows users to search these documents based on both keywords and geographic references. The process of supplying the documents to the GTS appliance for indexing is called ingestion. MetaCarta provides tools like the Share Crawler for ingesting documents. If these tools do not apply, you can write your own tools to ingest documents directly using this API. This guide defines the API's interface for document ingestion.

Concepts

Ingestion

Ingestion of a single document begins when the document is available for transfer to the appliance, and ends when the appliance acknowledges receipt of the document or returns an error. The acknowledgement does not mean that the document is yet searchable or that it contained any geographic content. Documents without geographic content are not indexed and thus not searchable by end users. The GTS appliance cannot determine a document will be searchable at the time of ingestion. Therefore, the acknowledgment should be considered only confirmation that the document has been properly received and will be further processed.

Document Identification and Re-Ingestion

Within GTS, documents are identified by unique URIs; the URI will be provided as the contact link for its document in the Web Search Interface. If there is no browser-based interface to the document, the URI for every document must still be unique.

If two documents are ingested with the same URI, the GTS appliance considers them the same document and the document ingested last will replace the previously ingested document. This process is called reingestion or updating, and can be used to make sure the GTS appliance contains the latest version of a document. When documents in the source repository change, they should be reingested to keep the GTS appliance up to date. The entire document must be reingested. The GTS appliance does not accept deltas.

Note: This replacement includes metadata; if a file is ingested as part of "Collection1" and then is later ingested as part of "Collection2," it will *only* show up in search results as part of Collection2. For more information on collection names and other metadata, please keep reading.

Document Size

The GTS appliance will accept files of any size. However, only the first 50MB of text in the document is indexed. You must still ingest the entire file as the 50MB limit is enforced after conversion to text. Thus, a 500MB PDF file that contains only 100K of text will be entirely indexed; but a 500MB text file will only have its first 50MB indexed.

Note: Your site administrator may have configured a maximum document size; please see page 15.

Appliance Capacity

There is currently no method to programmatically determine if there is enough space on the Appliance to ingest a document. If the appliance is out of space it will return an error instead of an acknowledgment on any attempt to ingest a document. For more information of managing the space available on the GTS appliance, see the *GTS Administrator's Guide*.

Metadata

Each document may have associated metadata. In this context, metadata is specifically referring to a named value. The metadata is carried along with the document throughout the system. Heavy usage of metadata will result in extra space usage on the GTS appliance. The maximum size of any metadata field is 250 characters for the metadata name and value fields combined.

Starting with MetaCarta GTS version 3.5, you can now have more than one value for a given metadata field.

Collection Name Metadata

The GTS Web Search interface allows users to choose a specific collection of documents to limit their searches. For example, a site may have documents from multiple publications; if the different publications have different collection name metadata, a user can limit her search to only one publication. The Ingestion API allows attaching this collection name metadata to documents.

You should not attach more than one piece of collection name metadata to the same document. If you do so, the document will be deleted if any of the named collections to which it belongs is deleted, and the document will not show up properly if a user searches in multiple collections at the same time.

Note: Collection Name metadata is passed to the appliance as standard metadata, but is handled specially by the GTS appliance.

Deletion

When documents have been ingested that need to be removed from the GTS appliance, you can delete them by URI through this API. This deletion occurs one document at a time. In addition, documents which have the same collection name metadata can be deleted as a group. The space used by deleted documents is not immediately available for more documents; there is a delay before the space is freed. However, the documents will immediately stop being returned in search results.

Document Templates

Each document may be provided with a document template which will limit which section of the document is indexed. The document template is defined in XML. For more information on document templates, see the *Document Templates Administrator's Guide*.

Access Control Lists (ACLs)

Each document may be provided with an Access Control List (ACL) that is used to limit which end-users will be shown which documents in their searches. To use ACLs, the GTS appliance must be configured to authenticate users. If the GTS is not configured to authenticate users, the inclusion of document ACLs will prevent all ACLed documents from being returned in users' searches.

Base64 encoding

When binary data needs to be transferred over text-only channels, there is a method called base64 encoding which will encode the data in such a way that no binary data need be transmitted. Base64 encoding is defined in RFC 2045 (<http://www.ietf.org/rfc/rfc2045.txt>) Base64 encoding is only necessary for ingestion if document templates or ACLs are needed.

HTTP

The Ingestion API for the GTS appliance is based on the HTTP protocol as defined in RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>) Both HTTP 1.0 and 1.1 connections are supported.

Streaming

When transferring large documents to the GTS appliance, it is important that neither the server or client be required to hold the entire document in memory. The reading and transfer of a document in smaller chunks is called streaming. The GTS Ingestion API supports this feature and will accept files of any size without undue memory usage.

Note: Streaming is different from HTTP chunked encoding.

Simultaneous Transfer

To maximize resource usage, the GTS Ingestion API will accept more than one document at a time. Currently, it will accept up to 10 documents simultaneously.

Authentication

The Ingestion API requires an authenticated connection. The authentication used is HTTP Basic Auth as defined in RFC 2617 (<http://www.ietf.org/rfc/rfc2617.txt>). The authentication channel is not considered secure unless you encrypt using SSL. For more information on creating a user for the Ingestion API and enabling SSL, see the “Basic Authentication” and “Secure Sockets Layer” sections of the *GTS Administrator’s Guide*.

Authorization

Any user created for authentication to the Ingestion API is implicitly authorized to upload and delete documents, and has full access to the API. API users are explicitly trusted to provide accurate ACLs on documents they provide for ingestion.

Interface Definition

Status

The basic method of invocation is an HTTP POST request. The URL to use in the request is:

```
http://<GTS-Appliance>/services/HTTPIngest?STATUS
```

unless SSL is enabled, in which case the URL would be:

```
https://<GTS-Appliance>/services/HTTPIngest?STATUS
```

If the HTTP Ingestion server is functioning properly, it will simply return “OK.” Otherwise, please see the Troubleshooting section on page 14.

Request Header: MetaCarta-Verbose-Response

This header is optional and may contain any data; the data is discarded. If this header is present, after reporting the status of the Ingestion interface, the appliance will also respond with two values, as follows:

```
STOPPED=False  
MaxContentLength=None
```

If ingestion has been stopped, the `STOPPED` variable will instead be set to `True`; if there is a maximum content length set by the administrator of the appliance, it will be displayed here. This status invocation may display more information in future releases; any new information will be displayed in the same `name=value` format.

Ingestion

The basic method of invocation is an HTTP POST request. The URL to use in the request is:

```
http://<GTS-Appliance>/services/HTTPIngest
```

unless SSL is enabled, in which case the URL would be:

```
https://<GTS-Appliance>/services/HTTPIngest
```

The document to be ingested is the payload of the HTTP POST request and any parameters regarding the document, such as metadata, ACLs, or the document's URI, are sent as HTTP Request headers. The following headers are available:

Request Header: Document-URI (required)

This header is required and contains the URI used to uniquely identify documents within the GTS appliance. This header must occur only once.

The provided URI must be valid according to RFC 2396 (<http://www.ietf.org/rfc/rfc2396.txt>). In particular, all special characters must be correctly escaped. See section 2 in the above RFC.

When the GTS user wishes to see an original document, the user will click on this URI in their web browser. The GTS GUI assumes that clicking on this URI from a web browser will present the user with the original document. In some cases, this will not work. If so, the appliance administrator should enable the cached link as described in the *MetaCarta Appliance Administration Guide* to make these documents available to search users.)

Request Header: Content-Length (required)

This header is required and contains the length in bytes of the POST data for this request. This length is identical to the size of the document to be ingested. Most HTTP libraries will automatically set this header. This header must occur only once.

Request Header: Document-Metadata (optional)

This is an optional header whose value is a name and value pair representing a single piece of metadata attached to the ingested document. The format for the value of this header is:

```
name=value
```

A metadata name may not contain the equals sign (=), leading or trailing whitespace, or the null character; its value may not contain leading or trailing whitespace, or the null character. Commas and backslashes must be escaped with a backslash (\, or \\).

This header may occur multiple times to set multiple metadata values. To set multiple values for a single metadata name, use multiple headers. To set a collection name, use the name "collection_name" in this header. We strongly recommend you do not set multiple collection names (see page 4).

Request Header: Document-ACL (optional)

This header is optional and contains a base64 encoded XML document which describes the Access Control List for the ingested document. This header must occur only once. The format of the XML and the semantics of the ACLs are defined in the “Building Collections with ACLs” section of the *GTS Administrator’s Guide*.

Request Header: Document-Template (optional)

This header is optional and contains a base64 encoded XML document which describes the template to apply to the ingested document. For more information on document templates, including the format and semantics of the XML document, see the *Document Templates Administrator’s Guide*. This header must occur only once.

Response

On success, the request will result in a 200 HTTP response with an HTTP payload containing the string `OK`. On error, an HTTP 400 code is returned and the payload will contain the string `ERROR`, followed by details on what went wrong. If there is a problem with the GTS appliance, an HTTP 500 code is returned.

Deletion

The basic method of invocation is an HTTP POST request with no content, or an HTTP GET request. The URL to use in the request is:

```
http://<GTS-Appliance>/services/HTTPIngest?DELETE
```

unless SSL is enabled, in which case the appropriate URL is:

```
https://<GTS-Appliance>/services/HTTPIngest?DELETE
```

The document to be deleted is passed in the Document-URI header. This is the only header required and considered. Only one document may be deleted in each request.

Request Header: Document-URI (required)

This header is required and contains the URI used to uniquely identify documents with the GTS appliance. It is equivalent to the header of the same name used during the ingestion request as defined on page 7. This header must occur only once.

Response

On success, the request will result in a 200 HTTP response with an HTTP payload containing the string `OK`. On error, an HTTP 400 code is returned and the payload will contain the string `ERROR`, followed by details on what went wrong. If there is a problem with the GTS appliance, an HTTP 500 code is returned.

Collection Deletion

The basic method of invocation is an HTTP POST request with no content, or an HTTP GET request. The URL to use in the request is:

```
http://<GTS-Appliance>/services/HTTPIngest?DELETE_COLLECTION
```

unless SSL is enabled, in which case the appropriate URL is:

```
https://<GTS-Appliance>/services/HTTPIngest?DELETE_COLLECTION
```

The collection to be deleted is passed in the Collection-Name header. This is the only header required and considered. All documents that were ingested with this value in the collection_name metadata field will be deleted.

Request Header: Collection-Name (required)

This header is required and contains the collection used to identify a group of documents on the GTS appliance. It references the collection_name field which is set in the optional Document-Metadata header during the ingestion request as defined on page 7. This header must occur only once.

Response

On success, the request will result in a 200 HTTP response with an HTTP payload containing the string `OK`. On error, an HTTP 400 code is returned and the payload will contain the string `ERROR`, followed by details on what went wrong. If there is a problem with the GTS appliance, an HTTP 500 code is returned.

Examples

GTS Name and Authentication for All Examples

All of the following examples assume that HTTP Basic Auth has been configured for a user named `fred` with a password of `ginger`. For more information on creating a basic auth user for use with the Ingestion API, see the “Basic Authentication” section of the GTS Administrator’s Guide.

All of the examples assume the GTS appliance is named `gts`.

curl

`curl` is a command line utility available for both Linux and Windows from <http://curl.haxx.se>. Since `curl` is an HTTP client and can set custom headers, you can use it to ingest and delete documents. For example, to ingest the contents of the file `foo.txt` as `file://somewhere/network/accessible/foo.txt` run the following command:

Note: The following commands should each be on one line; they are line-wrapped in this example for your convenience. To enter them on one line, remove the backslashes.

```
curl -HDocument-URI:file://somewhere/network/accessible/foo.txt \  
-HDocument-Metadata:collection_name="Name Of Collection" \  
-d@foo.txt http://fred:ginger@gts/services/HTTPIngest
```

To delete that document file run the command:

```
curl -HDocument-URI:file://somewhere/network/accessible/foo.txt \  
http://fred:ginger@gts/services/HTTPIngest?DELETE
```

Python

Python has HTTP and base64 libraries included in the base distribution. The code below will ingest the string “MetaCarta is located in Cambridge, MA” as a document with URI [urn:100234](#) (this URI is not a URL, but is still valid for ingestion) and an ACL as defined in the code. The code then deletes the same document.

Note: The ACL is base64 encoded XML.

```
#!/usr/bin/python  
  
import urllib  
import base64  
  
# Constants for each header name  
HDR_DOCUMENT_URI = 'Document-URI'  
HDR_DOCUMENT_METADATA = 'Document-Metadata'  
HDR_DOCUMENT_TEMPLATE = 'Document-Template'  
HDR_DOCUMENT_ACL = 'Document-ACL'  
  
document = "MetaCarta is located in Cambridge, MA"  
acl = """<document-acl>  
    <acl scope="file"><allow>S-1-1-0</allow></acl>  
    </document-acl>"""  
gts = 'http://fred:ginger@gts/services/HTTPIngest'  
  
# Create a URL opener and set the headers  
uo = urllib.URLopener()  
uo.addheader(HDR_DOCUMENT_URI, 'urn:100234' )  
#NOTE: newlines must be removed from the base64 output  
acl = base64.encodestring( acl ).replace( '\n', '' )  
uo.addheader(HDR_DOCUMENT_ACL, acl )  
  
# Connect and make the request  
try:  
    conn = uo.open( gts, data=document )  
  
    # Read and print the response  
    print conn.read()
```

```
# Now Delete in a similar fashion
uo = urllib.URLopener()
uo.addheader(HDR_DOCUMENT_URI, 'urn:100234' )
conn = uo.open( gts + '?DELETE' )
print conn.read()
except IOError, e:
    # Catch and print any HTTP Errors
    print e
    print str(e[3])
    # Print the local traceback
    raise
```

This file is also available in plain-text form as `HTTPIngestion.py` in the HTTP Ingestion Examples archive. For information on accessing files in this archive, please see page 14.

Please contact MetaCarta Support at support@metacarta.com for more help with ingestion using this language.

Java

Java has the `java.net.URLConnection` class, which can be used to send files to the appliance via HTTP. Alternately, here is sample code that uses Apache classes to talk to the Ingestion API:

```
import java.io.File;
import java.io.FileInputStream;
import java.net.URL;

import org.apache.commons.httpclient.Header;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.InputStreamRequestEntity;
import org.apache.commons.httpclient.methods.PostMethod;

/**
 * Example of how to use HTTPIngest API via Java.
 *
 * @author Geoffry Meek
 */
public class HttpIngestSample {

    public static void main(String[] args) throws Exception {
        if ((args.length != 5) && (args.length != 3)) {
            System.out.println("Usage: HttpIngest file uri ingest_url [username \
password]");
            System.out.println("<file> - full path to a file to be posted");
            System.out.println("<uri> - Document-URI used to uniquely identify \
the document in the GTS");
        }
    }
}
```

```
        System.out.println("<ingest_url> - URL pointing to the HTTPIngest \
service");
        System.out.println("<username> - HTTPIngest basic auth username");
        System.out.println("<password> - HTTPIngest basic auth password");
        System.exit(1);
    }

    /* create new HttpClient */
    HttpClient client = new HttpClient();

    /* use the POST HTTP method (GET should work as well) */
    PostMethod httppost = new PostMethod(args[2]);

    /* check to see if we should send Basic Auth credentials */
    try {
        if (args[3]!=null) {
            URL url = new URL(args[2]);

            client.getState().setCredentials(
                new AuthScope(url.getHost(), url.getPort()),
                new UsernamePasswordCredentials(args[3], args[4])
            );

            httppost.setDoAuthentication(true);
            client.getState().setAuthenticationPreemptive(true);
        }
        catch (ArrayIndexOutOfBoundsException e) {
            /* ignore this */
        }

        /* the Document-URI header is the unique identifier in the GTS for \
a given document.
        * The HTTPIngest service requires that it be set.
        */
        httppost.setRequestHeader(new Header("Document-URI", args[1]));

        /* create a new file, and set it as the request entity */
        File file = new File(args[0]);
        httppost.setRequestEntity(new InputStreamRequestEntity(
            new FileInputStream(file), file.length()));

        try {
            client.executeMethod(httppost);

            if (httppost.getStatusCode() == HttpStatus.SC_OK) {
                System.out.println(httppost.getStatusCode());
            } else {
                System.out.println("Unexpected failure: " + \
httppost.getStatusLine().toString());
            }
        } finally {
```

```
    httppost.releaseConnection();  
  }  
}
```

This file is also available in plain-text form as `HTTPIngestion.java` in the HTTP Ingestion Examples archive. For information on accessing files in this archive, please see page 14.

Please contact MetaCarta Support at support@metacarta.com for more help with ingestion using this language.

C/C++

Using C/C++, you have access to the library version of `curl` (see page 9).

Since the HTTP protocol is so simple, and only basic requests and responses must be made, developing directly against the operating system's sockets API is also an option.

Please contact MetaCarta Support at support@metacarta.com for more help with ingestion using this language.

C#/VB.NET (.NET)

The .NET platform provides the `System.Net.HttpWebRequest` class, which can be used for ingestion.

Please contact MetaCarta Support at support@metacarta.com for help with ingestion using this language.

JavaScript

Ingestion can be performed in JavaScript using the `XMLHttpRequest` object which is available in most modern browsers. However, browser security compartmenting and sandboxing may make this difficult to actually use.

Please contact MetaCarta Support at support@metacarta.com for help with ingestion using this language.

Visual Basic

From Visual Basic, using the `WININET.DLL/WINHTTP.DLL` interfaces or a related ActiveX control are recommended.

Please contact MetaCarta Support at support@metacarta.com for help with ingestion using this language.

Extracting the Examples Archive

To extract the examples archive, in the new directory `HTTPIngestionExtras`, to the current working directory, simply run the following command:

```
metacarta:~$ tar -xvzf /usr/share/doc/metacarta/HTTPIngestionExtras.tgz
```

You may find it convenient to just extract the entire archive into the `/usr/share/doc/metacarta/` directory. This will allow you to access all of the attached files at any time without having to expand them, but does take up additional disk space on the appliance. To do this, run

```
metacarta:~$ cd /usr/share/doc/metacarta
```

before running the command above.

Troubleshooting

Server responds with ERROR '...header required but not present'

Usually means a misspelled header. You might not be correctly passing the headers. Headers are case-sensitive, so "metadata" and "Metadata" are not the same thing.

Server responds with ERROR '...must be in the form name=value, not:...

When using the Document-Metadata header, this response means that the value of the header is incorrectly formatted.

Server responds with 'Your browser sent a request that this server could not understand.' / 'Size of a request header field exceeds server limit.'

The GTS appliance will only accept HTTP header lines of 32K or less in size. Sending header lines of larger size will result in the above error message. Most HTTP client libraries will automatically split up longer headers into separate header lines. However, if the library used does not do this automatically, the headers can be broken up into lines of 32K or less ending in a comma so the server will be able to build a single header out of the multiple header lines. The base64 encoded Document-ACL and Document-Template headers are the most likely headers to hit the 32K limit.

In versions 2.6.2 and higher, this limit has been increased from 8K, greatly reducing the likelihood of this error.

Server responds with 'false'

This generally means that the GTS appliance does not have sufficient space to process and index ingested files. You can confirm this with the Status command, defined on page 6. You or your administrator should free up more space on the appliance, either by deleting ingested files or by removing data from the appliance.

For more information about fixing this error, please see the Troubleshooting section of the *MetaCarta GTS Administrator's Guide*.

Server responds with error code 403

This error code means that the document has been rejected but you are free to send more documents. One possible cause is that the appliance administrator has enabled virus detection on the appliance, and the file sent had a virus in it. The file should not be resent unless it has been changed and you believe there is no longer a virus in it.

Alternately, this error could be because the administrator of your appliance has set a maximum limit on the size of files that can be ingested. You can determine this limit with the Status command, defined on page 6. This may be done to preserve network bandwidth, especially when using the Share Crawler. If you need to change this limit, you or your administrator should contact MetaCarta support at support@metacarta.com.

Encoding errors for ACLs and Templates

If the XML document for either ACLs or templates is incorrectly base64 encoded, or has invalid XML encoding, you will get this error. Common causes include forgetting to base64 encode the XML.

XML Errors for ACLs and Templates

The Ingestion API will validate the ACL and template XML documents. Any non-encoding XML format errors will contain the line and column of the error position in the response. You will then need to fix your XML and try reingesting the document.