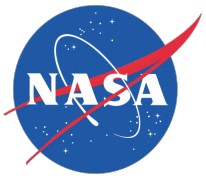
UTM

Unmanned Aerial Systems Traffic Management



Client Interface Document

DRAFT: Not for public release

NOTE: This document was sent directly to particular individuals and organizations directly from NASA Ames Research Center. If you received this document from another source, that source was not authorized to provide it. In that case, please delete all electronic copies of the document, refrain from printing it.

Version	Author	Date	Notes
0.1	Joseph Rios (Joseph.L.Rios@nasa.gov)	15 Aug 2014	Initial internal draft
0.2	Joseph Rios	10 Sep 2014	Draft for stakeholder review
0.3	JR	22 Oct 2014	Updates and Websockets

DOCUMENT OVERVIEW 3

CLIENT FUNCTIONS 4

Flight Submission 4

Flight Approval 4

Flight Decision Status Updates 4

Flight Position Updates 4

UTM Data Requests 5

Publication Services 5

OVERALL ARCHITECTURE 6

WFS 6

WFS-T 8
Flight Plan Submission 8
Flight State Updates 11

Publication Services 12
Client communication 12

AUTHENTICATION AND AUTHORIZATION 13

General overview 13

Layer name assignment 13

Credentials 14

Cross origin Resource Sharing (CORS) 14

Example usage 14

Document Overview

This document defines the interface to the prototype Unmanned Aerial Systems Traffic Management (UTM) system. This is a living document and will evolve as tests are conducted, requirements are gathered, and general lessons are learned. The intended audience of the information included herein includes NASA developers, client-side developers, and other interested stakeholders. The descriptions should allow for a client-side developer to implement software that interfaces with the prototype UTM system. The functionality provided by UTM to the client will also be detailed. This functionality is expected to grow as the UTM system grows, therefore the information provided here should not be interpreted as an indication of the all of the ultimately intended functionality of UTM.

Client Functions

Clients of the UTM system will be able to interact in specific ways the system. The functions and services provided to clients will be described in this section. The technical details of how to perform each of the functions are provided in a later section.

Flight Submission

One of the key functions of UTM will be to provide resource mitigation, contingency management, and system information to operators of UAS flights. To accomplish these UTM functions, operators will provide their planned flights to the UTM system. This flight information submission process is one of the key client-facing functions of the UTM system. These flight submissions should be considered “requests” for access to the airspace.

Flight Approval

In conjunction with the submission of flight information, UTM will provide clients with an initial decision on the status of their flight submission as soon as it is able to do so. Ideally this could happen synchronously with the flight submission, but realistically, it will need to be an asynchronous process to provide flexibility to the system in terms of current load, decision-making difficulty, and other potentially uncontrollable factors such as network reliability.

Flight Decision Status Updates

There may be situations where the initial flight approval decision needs to be updated. In this case, the UTM system will provide a notification to the appropriate operators of the update. Such updates would likely be due to updated information about the airspace (new weather information, equipment outages, uncooperative vehicle intrusion, etc.).

Flight Position Updates

Clients will be able to submit state information regarding their flights. Essentially this would be a data point representing where the flight is at a given time, perhaps together with other state information like heading, attitude, and other data. The client will provide these data in a manner similar to the submission of the flight plan. The early stages of UTM expect that the UAS operators will provide surveillance of their own aircraft and can communicate the appropriate state data via an Internet connection. In the future, the role of surveillance will likely be different and involve an additional entity or collaboration between operators and UTM maintainers.

UTM Data Requests

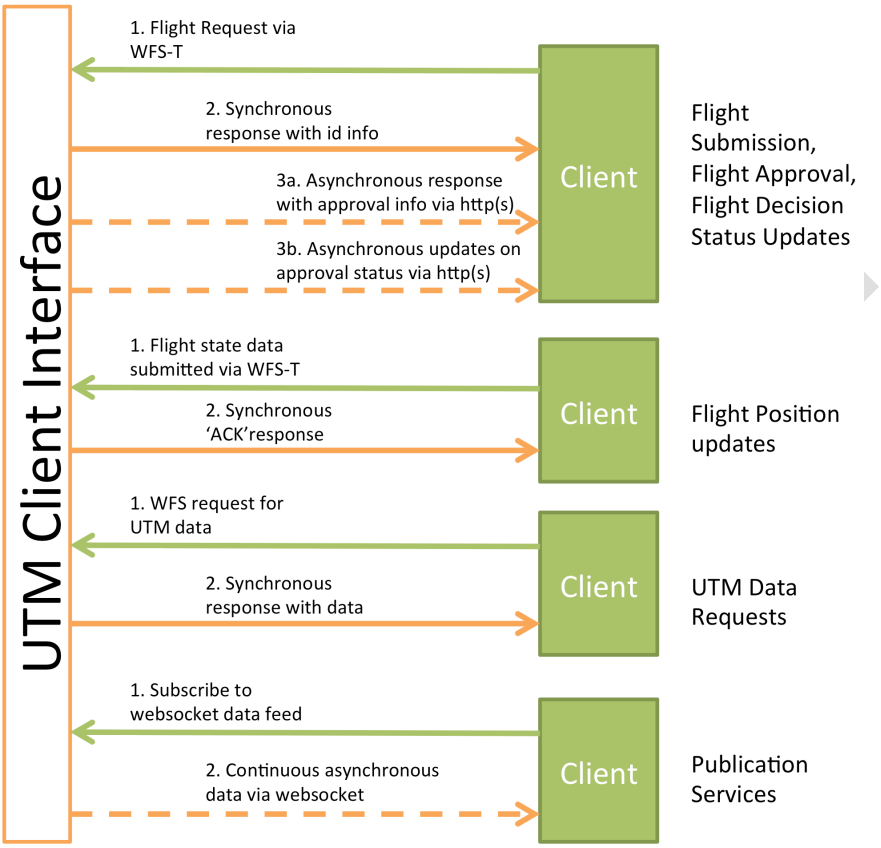
Given that UTM will have access to all flight plans in the system, UTM can provide information related to those plans to other users in the system. These flight data may be appropriately delayed, obfuscated, redacted, etc. as the operating concept for UTM dictates (yet to be determined). In general, the goal of this function to clients is to provide a complete of a picture as possible as to all of the flights in the system. This is in support of common situational awareness for all stakeholders involved in UTM.

Publication Services

UTM may also provide publication services of data relevant to the system. This concept would support a standard publish-subscribe model for data delivery. Clients may subscribe to specific or general data feeds that will continually supply updates about the system. These updates may include new flight plans that have been submitted or decided upon, constraints in the system, general information, etc.

Overall Architecture

Before detailing the client interface for implementation purposes, we provide a diagram and description to help ground the discussion.



WFS

The mechanism retrieve specific data from the UTM system is a Web Feature Service (WFS) request. WFS is an international standard for exchanging geospatial data defined by the Open Geospatial Consortium (OGC) and is completely described in documentation they produce. In this document we will only discuss WFS at a high level and provide some examples relevant to UTM. For a deeper understanding, the interested reader is directed to the relevant WFS documentation.

A compliant WFS server will describe its capabilities upon request. The current server for UTM is located at https://tmiserver.arc.nasa.gov/geoserver but that URL may change as the system matures. We will refer to that URL as

UTM_GEOSERVER_URL for the remainder of this document. To retrieve the capabilities of the UTM server, a client may send an HTTP(S) GET request to:

```
UTM_GEOSERVER_URL/wfs?request=GetCapabilities
```

This request will return an XML document detailing what the server can accomplish. One of the key pieces of information for UTM (and most WFS servers) are the Features that can be retrieved from the server. These are listed under the XML element "FeatureTypeList" as "FeatureType" elements. A sample from the current implementation (this will likely change throughout development) is as follows:

```
<FeatureType>
    <Name>utm:flights_guest</Name>
    <Title>flights_guest</Title><Abstract/>
    <Keywords>flights_guest,features</Keywords>
    <SRS>EPSG:4326</SRS>
  <LatLongBoundingBox minx="-1.0" miny="-1.0" maxx="0.0" maxy="0.0"/>
</FeatureType>
```

This FeatureType provides information about the "utm_operations" Feature. A client could then request further information about this Feature through a "DescribeFeatureType" request as follows:

```
UTM_GEOSERVER_URL/wfs?request=DescribeFeatureType&version=2.0.0&service
=WFS&typeName=utm:flights_guest
```

This request would then return specific information describing the "flights_guest" data element. Using that information, a client would be able to construct more advanced queries as well as Transactions to the WFS server to insert data. To request specific data, a client may again inspect the information returned from the GetCapabilities request. There are various filters supplied by the Feature Encoding Scheme (FES) supported by the server that include scalar parameter options such as "PropertyIsBetween," spatial operators such as "Intersects," temporal operators such as "After," amongst several others. Again, the WFS documentation would be the best resource to understand how to fully make use of these parameters.

The results may be returned in the format of the client's choice as supported by the server. As might be expected, the supported formats are described in the return from a GetCapabilities request to the server. Some more typical requests based on bounding boxes, counts, etc. for various return types might look like the following:

Select exactly 4 Features from flights_guest:

```
UTM_GEOSERVER_URL/wfs?request=GetFeature&version=2.0.0&typeName=utm:fli
ghts_guest&count=4
```

Select exactly 4 Features from flights_guest in JSON format:

```
UTM_GEOSERVER_URL/wfs?request=GetFeature&version=2.0.0&typeName=utm:fli
ghts_guest&outputFormat=application/json&count=4
```

Select a specific feature by its feature ID in CSV format:

```
UTM_GEOSERVER_URL/wfs?request=GetFeature&version=2.0.0&typeName=utm:fli
ghts_guest&featureid=flights_guest.1&outputFormat=csv
```

Requests for data via WFS are appropriate for retrieving data relevant to a client's needs. For example a client may want to see all planned operations for a specific time period in a specific location. An appropriate request to the WFS server can return this information in the format of the client's choosing (as supported by the server). A client may be a basic mapping application and as the view of the map changes, the client may request appropriate data for the new view area. Other use cases will develop along with the UTM system.

WFS-T

The WFS server that has been implemented for the prototype UTM system supports WFS Transactions (WFS-T), which allows clients to submit data to the UTM system. As with basic WFS, WFS-T is fully specified by the OGC and complete documentation is available elsewhere, so we only provide a brief overview here.

WFS-T within the UTM architecture is appropriate for the submission/updating of flight plans and the update of flight state data (e.g. a flight's current position, heading, speed). The UTM system may then use this information for approving or denying submitted plans based on system constraints. The data may also be used to keep other users of the UTM system aware of the various operations in the system. These data may represent current, planned, or completed operations.

Since the volume of data sent to the server may be large and cumbersome to include in a URL, all WFS-T communications to the server are completed with an HTTP(S) POST of XML data to the server.

Flight Plan Submission

Currently we have only implemented the ability of a client to submit a flight plan consistent with our database schema. There is a sample snippet in the "Authentication and Authorization" section (again, **the specific data elements will likely change in the near future and often**, but the process will remain mostly the same):

This would generate a valid record in the UTM operations table that could then be retrieved by other users in the system. This submission of flight data is a "request" that would not immediately enter the UTM system as an approved operation. The plan will be inspected for validity, safety, and other constraints. The UTM system or UTM manager would then decide if the operation was to be approved or denied. This process will be described more completely in another, future document.

The client is supplied a synchronous response to a successful WFS-T request. That response will contain an operation ID that the client can use for future communication with the UTM system. An example of this response is provided below:

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:WFS_TransactionResponse version="1.0.0"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs
http://localhost:8080/geoserver/schemas/wfs/1.0.0/WFS-transaction.xsd">
       <wfs:InsertResult>
              <ogc:FeatureId fid="flight_guest.10”/>
       </wfs:InsertResult>
       <wfs:TransactionResult>
              <wfs:Status>
                     <wfs:SUCCESS/>
              </wfs:Status>
       </wfs:TransactionResult>
 </wfs:WFS_TransactionResponse>
```

It is expected that the operator submitting the plan will also supply a callback URL that would be used by the UTM system to POST a decision on the flight back to the client. The clients would be responsible for servicing that HTTP(S) POST to the URL that they supplied. The data communicated via this channel will be in JSON format. The draft schema for the decision is provided below. In the future, this schema will be available at a static URL for use by clients in validation of the responses and for developing their processing capabilities for the response.

```
{
  "$schema": "http://json-schema.org/schema#",
  "title": "UTM Decision Message Schema",
  "type": "object",
  "properties": {
    "flight_id": {
      "type": "string"
    },
    "decision": {
      "type": "string"
    },
    "reason": {
      "type": "string",
      "optional": true
    },
    "constraints": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "table": {
            "type": "string"
          },
          "id": {
            "type": "string"
          }
        },
        "additionalProperties": false
      },
      "optional": true,
      "additionalProperties": false
    }
  }
}
```

The same JSON schema written more compactly in the “orderly” format (http://orderly-json.org/):

```
object {
  string flight_id;
  string decision;
  string reason?;
  array [object {stringtable;stringid;}] constraints?;
}* `{
  "$schema": "http://json-schema.org/schema#",
  "title": "UTM Decision Message Schema"
}`;
```

In the event that the client does not or cannot receive the informational POST message regarding its requested flight (due to a bad URL, requestor’s server is down, no URL supplies, etc.), it would be the client’s responsibility to query the system for updated information regarding the requested flight. This means the

information exchange would be a “pull” from the client side rather than a “push” from the server to the client. Overall, this sort of polling for data is inefficient, thus the goal on the client side should be to develop robust enough mechanisms to handle the POST response from the UTM system. Using a client capable of handling POSTs from the UTM system may become a requirement of the UTM system as a whole.

Flight State Updates

The client will use a similar WFS-T request/submission for flight updates. As state information is available, the client may submit that information to a position table in the database that references the flight operations table via the operation id supplied in the original flight request submission. Note that this would note require a callback URL supplied by the client. The “SUCCESS” message returned by the WFS server will be the only acknowledgement of a flight update submission. A sample using the current UTM schema is provided below. Recall that the specific data elements will change soon and potentially often as development on the system proceeds. Any clients that are built should be designed with the appropriate flexibility to handle changes in the data elements and layer names. The goal is to design clients that can handle generating valid WFS-T requests.

```
<wfs:Transaction service="WFS" version="1.0.0"
  xmlns:topp="http://www.openplans.org/topp"
  xmlns:ogc="http://www.opengis.net/ogc" xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:utm="http://opengeo.org/utm"
  xsi:schemaLocation="http://www.opengis.net/wfs
                      http://schemas.opengis.net/wfs/1.0.0/WFS-transaction.xsd
                      http://opengeo.org/utm">

    <wfs:Insert>
    <utm:enroute_positions_{operator}>
     <utm:flight_id>9999</utm:flight_id>
     <utm:heading>213</utm:heading>
     <utm:location>
        <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
           <gml:coordinates xmlns:gml="http://www.opengis.net/gml" decimal="."
cs="," ts=" ">0.0,0.0</gml:coordinates>
        </gml:Point>
      </utm:location>
     <utm:speed>101</utm:speed>
    <utm:time>2014-10-09T17:35:21</utm:time>
  </utm:enroute_positions_{operator}>
  </wfs:Insert>
</wfs:Transaction>
```

Publication Services

In the future, UTM may provide data feeds for certain data stored within UTM. One might consider a subscription to data relevant to a given area, or a given operator, or cancelled flight plans, or any number of other specific (potentially client-defined) data streams. These publication services will be the method for supplying updated flight track information to clients. As an initial implementation, the UTM system will use websockets to provide these live feeds.

The flight track data is provided in 5-second intervals although this number is configurable on the server side and may change in the future. The data provided are the most recent track for each active flight in the system. Note that if multiple track updates are received from a particular aircraft in between the 5-second intervals, only the latest one will be submitted to users via the websocket connection. If more detailed track data are required, clients should request the appropriate data via the standard WFS request.

The data can be obtained from the following URL

<NOTE: This service may be unavailable for significant periods during development. If you are trying to test anything against this URL/service, let us know and we'll provide status information as to availability>
http://tmiserver.arc.nasa.gov/FlightTracker/tracker

The access to this URL is protected and hence it is required for the users to supply their user credentials.

Results are supplied in JSON format:

```
{"flightId":"1",
"lat":-63.060318,
"lon":56.432045,
"alt":234.0,
"speed":45,
"heading":100}
```

Client communication

SockJS clients can be used to connect to the server and obtain track information. Here is an example:

```
var socket;

function startClient() {
    console.log ("opening socket");
//Start the websocket client
   socket = new SockJS
("http://user:password@tmiserver.arc.nasa.gov/FlightTracker/tracker");
    //When the connection is opened, login.
    socket.onopen = function() {
```

```
      console.log("Opened socket.");
      socket.send("hello");
    };

    //When a message is received, parse it and do something with    it.
   socket.onmessage = function(a) {
      //process the message here
      console.log("received message: " + a.data);
      var message = JSON.parse(a.data);
    }
    socket.onclose = function() { document.write("Closed socket."); };
    socket.onerror = function() { document.write("Error during
transfer."); };

  }
```

In the current implementation, all registered users will receive flight track data that they are authorized to view, however in the future, users can subscribe to selective information that they would like to receive.

Authentication and Authorization

In this section we outline the authentication and authorization scheme of the prototype UTM system.

General overview

The UTM server obtains credentials from a locally installed and maintained Lightweight Directory Access Protocol (LDAP) application to authenticate and authorize users. UTM provides layer-level security. This means that an operator is allowed to read and write data only to layers upon which they are authorized. The current implementation of UTM has a publically (i.e. no authentication required) readable guest layer for which all authenticated users are authorized to write. In the future, it is expected that nearly all data provided to the UTM system will be publically readable to enhance common situational awareness amongst all stakeholders. The current implementation offers tighter layer-level security to allow partners an extra degree of security and privacy when needed.

Layer name assignment

Each UTM operator will be an assigned string that will uniquely identify that operator. This string will be used to name the layers that relate to that operator's operations. That string may correspond to the operator's name. For example if company "Fast Flying UAS, Inc." is a partner with UTM, they may be assigned the string "ffu." It should be noted that the layer names are protected from reading and writing from unauthorized users, but their existence is not a secret and is freely available via a "GetCapabilities" command to the WFS server. Thus, if a partner wishes to be more discreet in its operations during the initial stages of UTM development, the assigned string can be anonymized in some way. For example, "Fast Flying UAS" may be assigned the string "shhh_secret" so that their company will not be readily identifiable via the layer names.

The table below provides a summary of the layers relevant to user-by-user layer security.

Layer Name	Privilege	Purpose	Publically readable?
flight_submitted_{operator}	R/W	Contains all requested plans in 'Pending' status for {operator}	N
flight_{operator}	R	Contains all "approved" operations for {operator}	N
flight_submitted_guest	R/W	Contains all requested plans in 'Pending' status	Y
flight_guest	R	Contains all "approved" operations	Y

Credentials

At this point user registration will be a manual process. Operators will be provided with username, password, and operator ID (for use in layer naming as described above) that is required to when posting data to UTM. Each participating operator will only be issued one username and password to be shared within their organization. This is simply to limit IT administration on the UTM server. In the future, all individual users of the system will have their own credentials that are tied to their organization. So, if there are 8 users at Fast Flying UAS, they will eventually each get their own credentials that would authorize them to write data related to operations for Fast Flying UAS. In the initial months, all 8 users will share the single set of credentials supplied to Fast Flying UAS.

Cross origin Resource Sharing (CORS)

UTM servers have been "CORS" enabled in order to serve browser-based clients. Due to security restrictions, the clients will have to "register" their domain name and port of origin (location from which request is made) with the UTM server.

Example usage

The UTM application requires user authentication in order to GET and POST data to UTM layers. Hence clients accessing UTM data will need to supply their user credentials in the HTTP headers of their requests.

The following is a sample GET query from the command line. This query will return all approved flights for {operator} that are scheduled to begin after 00:00 UTC on July 11th, 2014. To specify further temporal and spatial filters, the user is directed to the appropriate Geoserver and OGC documentation.

```
curl -v -u username:password -G
http://tmiserver.arc.nasa.gov/geoserver/wfs?service=WFS&version=1.0.0&request=GetFeature&typeName=utm:flights_{operator}&CQL_FILTER=(effective_time_begin%20AFTER%202014-07-11T00:00:00Z)&outputFormat=application/json
```

Please note that the typeName specifies the operator's layer name or any layer to which the operator has read privileges.

Sample POST to request for an approval of an operation plan:

```
<wfs:Transaction service="WFS" version="1.0.0"
  xmlns:topp="http://www.openplans.org/topp"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:wfs="http://www.opengis.net/wfs"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:utm="http://opengeo.org/utm"
  xsi:schemaLocation="http://www.opengis.net/wfs
                      http://schemas.opengis.net/wfs/1.0.0/WFS-transaction.xsd
                      http://opengeo.org/utm">

  <wfs:Insert>
   <utm:flights_submitted_{operator}>
     <utm:operator>Random</utm:operator>
     <utm:user_id>14</utm:user_id>
     <utm:operator_id>9</utm:operator_id>
     <utm:unmanned>true</utm:unmanned>
     <utm:state>1</utm:state>
     <utm:aircraft_type>Black Hawk 2</utm:aircraft_type>
     <utm:primary_contact_name>Random Person</utm:primary_contact_name>
     <utm:submit_time>2014-07-30T13:00:00Z</utm:submit_time>
     <utm:effective_time_begin>2014-08-18T16:00:00Z</utm:effective_time_begin>
     <utm:effective_time_end>2014-08-18T21:00:00Z</utm:effective_time_end>
     <utm:primary_phone>111-111-1111</utm:primary_phone>
     <utm:notes>Some interesting info 2 </utm:notes>
     <utm:notam_station>ZAN 06/032</utm:notam_station>
     <utm:flight_geography>
        <gml:LineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
          <gml:coordinates decimal="." cs="," ts=" ">
            -147.6123046875,65.07213008560697
           -147.1728515625,64.18724867664994
            -149.4140625,64.66151739623561
           -147.6123046875,65.07213008560697
         </gml:coordinates>
        </gml:LineString>
     </utm:flight_geography>
```

```
      <utm:valid>true</utm:valid>
      <utm:region>true</utm:region>
    </utm:flights_submitted_{operator}>
  </wfs:Insert>
</wfs:Transaction>
```

Command to post this content from a file called *insert_flight_operations_utm.xml* to UTM application:

```
curl -v -u username:password -XPOST –d
@insert_flight_operations_utm.xml -H "Content-type: application/xml"
UTM_GEOSERVER_URL/ows
```