Wedgetail API

Contents

1. General Information

1.1 Introduction

This document describes a programmatic interface for the wedgetail system, which allows local clients and third-party systems to exchange messages and query the server.
It is based on the REST principles (Represetational State Transfer)
More information on REST is at
http://en.wikipedia.org/wiki/Representational_State_Transfer
Libraries are generally available to allow RESTful services for most programming languages, including DOT NET, Silverlight, php and Ruby.

1.2 Underlying protocol stack

Connections are made to the server using HTTPS. It is assumed the client knows the DNS address of the server. The port is assumed to be 443 unless specifically arranged otherwise.

For each API function this standard specifies the absolute path within the server, thus forming a URI with the server name.

Where the path provided coontains "XXX" or "YYY" these should be substituted with specific data elements as defined for that function.

The standard also specifies the HTTP verb to use (GET or POST) and the MIME-type of the documents that are sent and received.

For most functions the document to be sent is in the type "application/xml', and similarly application/xml is returned by wedgetail

If the field is defined as "a date", it will be in the form yyyy-mm-dd.

1.3 Database structure

Logically the wedgetail database is a large flat collection of documents. Documents are of varying MIME types and are also tagged by creation date, associated patient, user who uploaded them, and logical type. Documents are immutable once created.

Documents may exist on various servers, synchronisation functions allow all the servers to maintain a single list of patients and a single index of documents, the documents themselves exist on only one server on the network.

Patient records are mutable, in the sense that a new record with the same patient ID can be created and made available for synchronisation, this replaces the old. The newest patient record is always the current one.
However servers can still use old records for searching, such as for maiden names.

"logical type" refers to their *clinical* role and is independent of their acutal format.
Document types are listed below

1.4 Security & Authorisation

Cryptographic security is provided by the SSL/TLS layer of HTTPS.
Clients authenticate using HTTP Basic authentication [5] for all connections.

1.4.1 Users

Clients should be aware that access to patient files, including appearance in search results, is dependent upon the server's authorisation model and the access granted to that specific user. Intentionally, clients will not be able to distinguish between when a patient is non-existent, and when they do not have sufficien taccess for the operation in question.

1.4.2 Patients

It is also possible to authenticate as a patient. The username is the wedgetail patient ID and the password is the patient password. Patient's access privileges are generally limited to documents about them and updating their own demographic details.

1.5 Patient Identification

1.5.1 Wedgetail Numbers

Patients are uniquely identified by the Wedgetail number, which is assigned by the system on patient creation.
After patient creation, clinical information cannot be added to the record until a consent from is received by the Wedgetail Adminsitrator (the Big Wedgie), and the uniqueness of that patient record has been established.

The IHI (Individual Health Identifier) will be mapped to (or replace) the Wedgetail number when it is implemented.

1.5.2 Local IDs

Clients connecting to the Wedgetail server can identify patients through the ID used by the local system, once these have been mapped to the patients wedgetail number.
The mapping is between the Wedgetail Number, LocalID, and the User's team.
Therefore clients connecting to Wedgetail need to ensure that the LocalIDs used are unique within their local system only.

1.6 Error Messages

The Wedgetail API can return a number of error messages.
These are listed in the appropriate sections below.
Note that malformed XML schema posted in the request will return a generic Internal Server error.

2.The Patient Resource

2.1. Creating a New Patient

Authentication via Basic HTTP Authentication

Note that new patient records will not be accessible (will remain 'unhatched') till their consent form is received by the Big Wedgie and details are confirmed (hatched)

POST to https://wedgetail.org.au/patients

XML format

<patient>
* <family_name>Potter</family_name>
* <given_names>Harry</given_names>
<known_as></known_as>
<address_line>Gryffindor House</address_line>
<town>Hogwarts</town>
<state>NSW</state>
<postcode>2477</postcode>
<sex>1</sex> (1=Male 2=Female)
* <dob>1994-11-01</dob>
<medicare></medicare>
<dva></dva>
<password>passwd</password>
**<localID>12345</localID>
</patient>

* required fields

** a local ID, if supplied, will be mapped to the new patient's wedgetail number.

Once mapped, the localID can be used instead of the wedgetail number for all XML calls.

RETURNS

```
<?xml version="1.0" encoding="UTF-8"?>
<message>Patient Created</message>
<patients>
  <patient>
    <family_name>Potter</family_name>
    <given_names>Harry</given_names>
    <dob>1994-11-01</dob>
    <wedgetail>HDvjskf</wedgetail>
    <medicare></medicare>
    <address_line>Gryffindor House</address_line>
    <town>Hogwarts</town>
  </patient>
</patients>
```

ERRORS

Error 01: Patient with that localID already created

If the localID has already been mapped to a different patient, Error 01 is generated.

The XML details of the patient that had previously been mapped is also returned

Error 01: Patient with that localID already created.

```
<?xml version="1.0" encoding="UTF-8"?>
<message>Error 01: Patient with that localID already created</message>
<patients>
  <patient>
    <family_name>Potter</family_name>
    <given_names>Harry</given_names>
    <dob>1994-02-09</dob>
    <wedgetail>HDvjskf</wedgetail>
    <medicare></medicare>
    <address_line>Gryffindor House</address_line>
    <town>Hogwarts</town>
  </patient>
</patients>
```

Error 02: Patient possibly already created

If the patient details (Family Name, DOB) appear to match one or more patients already registered with Wedgetail, “Error 02: Patient possibly already created" is returned. The XML contains the list of possible matching patients already created.

```
<?xml version="1.0" encoding="UTF-8"?>
<message>Error 02: Patient possibly already created</message>
<patients>
  <patient>
    <family_name>Potter</family_name>
    <given_names>Harold</given_names>
    <dob>1994-02-09</dob>
    <wedgetail>HD8p2rq</wedgetail>
    <medicare></medicare>
    <address_line>Gryffindor House</address_line>
    <town>Hogwarts</town>
  </patient>
  <patient>
    <family_name>Potter</family_name>
    <given_names>Ronald</given_names>
    <dob>1994-02-09</dob>
    <wedgetail>HDsafve</wedgetail>
    <medicare></medicare>
    <address_line>Diadem Alley</address_line>
    <town>London</town>
  </patient>
 </patients>
```

If Error 2 is generated, clients should ask the user if one of the returned patients matches the patient they are trying to create.

If so, you can use the ‘localmap’ service to map a localID to the patient wedgetail number. (see below)

If not, and the patient is unique, you can repost the command, but include in the XML of patient attributes the attribute **<force>true</force>**

The administrator will ultimately confirm the uniqueness of the new patient.

CURL example

curl -H 'Accept: application/xml' --user 'username:password' -H 'Content-type:application/xml' -k https://wedgetail.org.au/patients -d "<patient><family_name>Lembke</family_name><given_names>Will</given_names><known_as>Will Will</known_as><address_line>81 Dees Lane</address_line><town>Uralba</town><state>NSW</state><postcode>2477</postcode><sex>1</sex><dob>1999-02-09</dob><medicare></medicare><dva></dva><password>oddie</password></patient>" -X POST

2. 2 Searching for a Patient

GET to https://wedgetail.org.au/patients.xml

Authentication via Basic HTTP Authentication

Optional Search parameters are family_name, given_names, dob (format:1999-09-04)

Note that only details about patients that the user is authorised to see will be returned.

egs

https://username:password@wedgetail.org.au/patients.xml?family_name=weasley&dob=1994-03-31

https://username:password@wedgetail.org.au/patients.xml?family_name=weasley

returns

```
<?xml version="1.0" encoding="UTF-8"?>
<patients>
  <patient>
    <family_name>Weasley</family_name>
    <given_names>Ron</given_names>
    <dob>1994-03-31</dob>
    <wedgetail>Hdemosk56y</wedgetail>
  </patient>
  <patient>
    <family_name>Weasley</family_name>
    <given_names>Ginny</given_names>
    <dob>1996-03-02</dob>
    <wedgetail>Hdemo75wvc</wedgetail>
  </patient>
</patients>
```

Curl Example

curl -H 'Accept: application/xml' --user 'username:password' -H 'Content-type:application/xml' -k https://wedgetail.org.au/patients.xml -d "<family_name>Weasley</family_name>" -X GET

3. Mapping and the Localmaps resource

Users can ‘map’ local database IDs to wedgetail numbers, as discussed in section 1.5.2

3.1 Creating a mapping

To create a map between a given localID and a known wedgetail number, POST to the localmaps resource.

POST to https://wedgetail.org.au/localmaps

XML Schema

```
<localmap>
   <localID>27</localID>
   <wedgetail>HDaaaab</wedgetail>
</localmap>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<localmap>
  <message>Localmap created</message>
  <localID>30</localID>
  <wedgetail>HDaaaad</wedgetail>
</localmap>
```

Errors

1. Error 1: That wedgetail number is already mapped to another localID

```
<?xml version="1.0" encoding="UTF-8"?>
<localmap>
  <message>Error 1: That wedgetail number is already mapped to another localID</message>
  <localID>29</localID>
  <wedgetail>HDaaaaa</wedgetail>
</localmap>
```

2. Error 2: That localID is already mapped to another wedgetail number

```
<?xml version="1.0" encoding="UTF-8"?>
<localmap>
  <message>Error 2: That localID is already mapped to another wedgetail number</message>
  <localID>27</localID>
  <wedgetail>HDaaaaa</wedgetail>
</localmap>
```

3.2 FInding a mapping

Authentication Required
GET to https://wedgetail.org.au/localmaps.xml
eg https://wedgetail.org.au/localmaps.xml?localID=30

Returns

```
<?xml version="1.0" encoding="UTF-8"?>
<message>Found</message>
<localmap>
  <localID>30</localID>
  <wedgetail>HDaaaad</wedgetail>
</localmap>
```

4. The Narrative Resource

Authentication via Basic HTTP Authentication
Narratives are linked to patients via the patient's unique wedgetail number.

POST to https://wedgetail.org.au/narratives.xml

XML Format

```
<narrative>
        *<wedgetail>Hdemoc4z7v</wedgetail>
        <narrative_type_id>1 </narrative_type_id>   (see below for Type Index)
        <title>Health Summary</title>
        <content>This is where the content goes as text or rtf</content>
        <content_type>text/plain or text/rtf</content_type>
        <narrative_date>2009-04-05</narrative_date>
</narrative>
```

* NB <localID>XXXX</localID> can be used instead of <wedgetail>XXX</wedgetail> if a mapping has been established.

Our current Narrative Types are :

1 Health Summary
2 Medication Chart
3 Discharge Summary
4 Encounter
5 Allergies
6 Scratchpad
7 Results
8 Letter
9 Immunisations
10 Form
11 Prescription
12 Diagnosis
13 Care Plan

Return

```
<?xml version="1.0" encoding="UTF-8"?>
<message>Narrative created</message>
<narratives>
  <narrative>
    <wedgetail>HDaaaad</wedgetail>
    <narrative_type_id>1</narrative_type_id>
    <title>Health Summary</title>
    <content>This is where the content goes as text or rtf</content>
    <content_type>text/plain</content_type>
    <narrative_date>2009-04-05</narrative_date>
  </narrative>
</narratives>
```

ERROR Messages

The <message> field will contain an error message if the wedgetail number or localID do not map to a patient.

5 Results Service

The steps in the result service are

a) At the time of generating a pathology request for which you would like the subsequent result uploaded to Wedgetail, send an identifer for that request (request_set) to wedgetail, and receive an identifying 'ticket' in return.

The request_set identifer should be unique to that result and to that practice.

The ticket is a string that the patient will use to later retrieve their result, and is given to the patient.

b) When a new batch of results have subsequently been received and processed, check with Wedgetail to see which request sets have been registered at the time of requesting and therefore should be uploaded.

c) upload those results and actions

d) the patient enters their identifer on the server to view relevant uploaded results

NOTE: Deindentifaction of all information sent to Wedgetail is the responsibility of the client program that is using the service.

DO NOT send identifying features with your result uploads.

DO NOT use this service is this is unacceptable or not possible for you.

5.1. Upload a request set and ticket identifier to wedgetail.

POST to http://wedgetail.org.au/result_tickets

No authorisation required

At the time a pathology request is generated by the client program, the request identifer is 'registered' with wedgetail.

A ticket identifer (used by the patient) can be nominated or generated automatically by wedgetail.

An error message is returned if you allocate a ticket that is used by a different request_set.

XML

```
<result_ticket>
  <request_set>666666</request_set>
  <ticket>010102</ticket> (optional)
</result_ticket>
```

If the ticket identifier is not included, Wedgetail will return the correct ticket identifer, if the request set has already been registered, or will generate and return a new unique ticket identifier if not already registered.

The full result_ticket with ticket identifier is returned.

```
<?xml version="1.0" encoding="UTF-8"?>
<result_ticket>
  <request_set>666666</request_set>
  <ticket>010102</ticket>
</result_ticket>
```

Curl Example

curl -H 'Accept: application/xml' -H 'Content-type:application/xml' -k http://localhost:3000/result_tickets -d "<result_ticket><request_set>666666</request_set></result_ticket>" -X POST

5.2 Determine which request sets have been registered with wedgetail (and therefore have an associated ticket)

POST to http://wedgetail.org.au/result_tickets/check

No authorisation required

XML

```
<result_ticket_list>
        <request_set>213101</request_set>
        <request_set>666666</request_set>
</result_ticket_list>
```

Returns those request_sets in the request list which have been previously registered with Wedgetail.

```
<?xml version="1.0" encoding="UTF-8"?>
<result_ticket_list>
  <request_set>666666</request_set>
</result_ticket_list>
```

Curl example

curl -H 'Accept: application/xml' -H 'Content-type:application/xml' -k http://wedgetail.org.au/result_tickets/check -d "<result_ticket_list><request_set>213101</request_set><request_set>666666</request_set></result_ticket_list>" -X POST

5.3 Upload a set of actions and results

POST To http://wedgetail.org.au/actions

No authorisation required

XML

```
<action_list>
        <action>
                <request_set>iii</request_set>
                <name>FBC</name>
                <action_code>1</action_code>
                <comment>You can put as much text here, and it can be in plain text or rtf</comment>

                <identifier>3</identifier>
        </action>
        <action>
                <request_set>www</request_set>
                <action_code>1</action_code>
                <comment></comment>
                <identifier>3</identifier>
        </action>
</action_list>
```

Curl example

curl -H 'Accept: application/xml' -H 'Content-type:application/xml' -k http://wedgetail.org.au/actions -d "<action_list><action><request_set>iii</request_set><name>FBC</name><action_code>1</action_code><comment>Get found</comment><identifier>3</identifier></action><action><request_set>www</request_set><action_code>1</action_code><comment>Get found</comment><identifier>3</identifier></action></action_list>" -X POST