Gemini

API Usage Guidelines

Author (for this version):	National Grid IS
Owner:	National Grid O&T
Document Reference:	4026
Version:	3
Status:	~~For Representation~~For Approval
Date:	~~07 September 2005~~29 September 2005

nationalgrid

Table of Contents

1 Introduction

An Application Programming Interface (API) based alternative to screen access is provided with Gemini. Business Associates (BAs) of National Grid can use these APIs to access specific functions of Gemini. APIs provide an alternative to the historical use of screen scraping to create automated interfaces to the RGTA and AT-Link systems that Gemini replaces.

APIs are split into two basic categories:

- Functional APIs. These implement the Gemini business processes that are accessible via APIs.
- Session management APIs. These do not implement business processes. These are the login, logout and change password APIs.

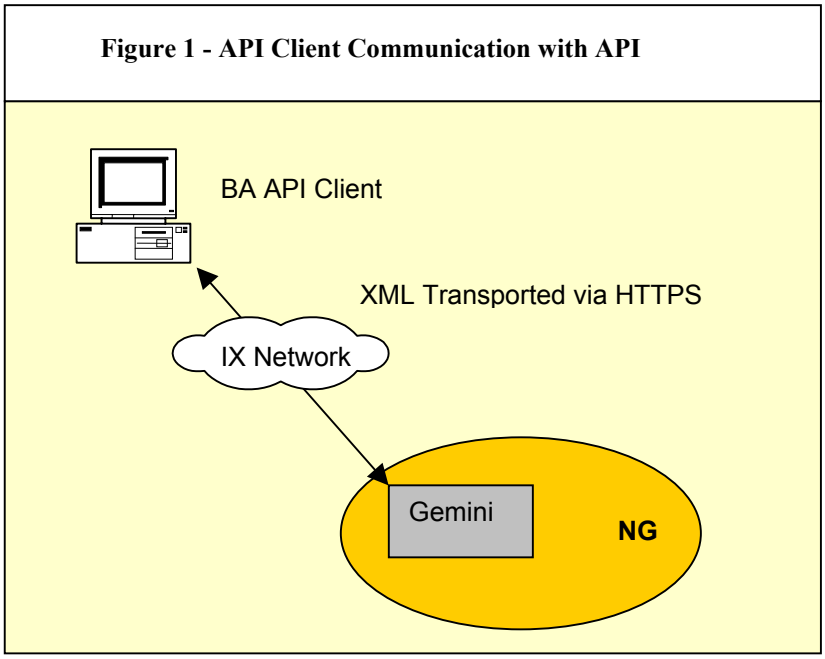
This document contains guidelines on how to use APIs. It also contains specifications for session management APIs (see section 3.1 Login API, section 3.3 Logout API and section 3.4 Change Password API). It does **not** contain specifications for the business functionality APIs that are implemented (see section 5 – Functional API Specifications). Functional APIs are specified in separate documents, one for each API.

To access APIs provided by National Grid, BAs must develop API client programs (henceforth shortened to API clients). This document provides guidelines for developing API clients.

Note that API client to API interaction does **not** use Citrix technology. This is reserved for screen based communication with Gemini.

1.1 *API Technology Overview*

Figure 1 - API Client Communication with API



BAs must develop API clients in order to use the APIs provided by Gemini. Guidelines for developing API clients are provided in section 2 – API Client Development Guidelines.

API clients specify a URL to access an API. These URLs are different from those of Gemini screens. API clients must issue an HTTPS request that contains API input parameters in XML format. After processing the request Gemini will return an HTTPS response with the output data, also in XML format, in the body.

Each API interaction is therefore a request/response pair. Communication between BAs and Gemini for API access is via the IX network.

The core technologies used to interact with Gemini APIs are HTTP[1] and XML. HTTP is used as the communications protocol. XML is used to represent the data that must pass between the API client and the Gemini APIs to invoke the relevant business functionality. **NB.** No XML is exchanged with the session management APIs, since they do not implement business functionality.

It is **not** our intention to extensively document and explain these technologies. They are not specific to Gemini APIs and are extensively documented elsewhere. See, for example, the documentation relating to HTTP, XML and XML Schemas published by the World Wide Web Consortium.

We will only explain aspects of these technologies that are especially pertinent to API client interaction with Gemini APIs.

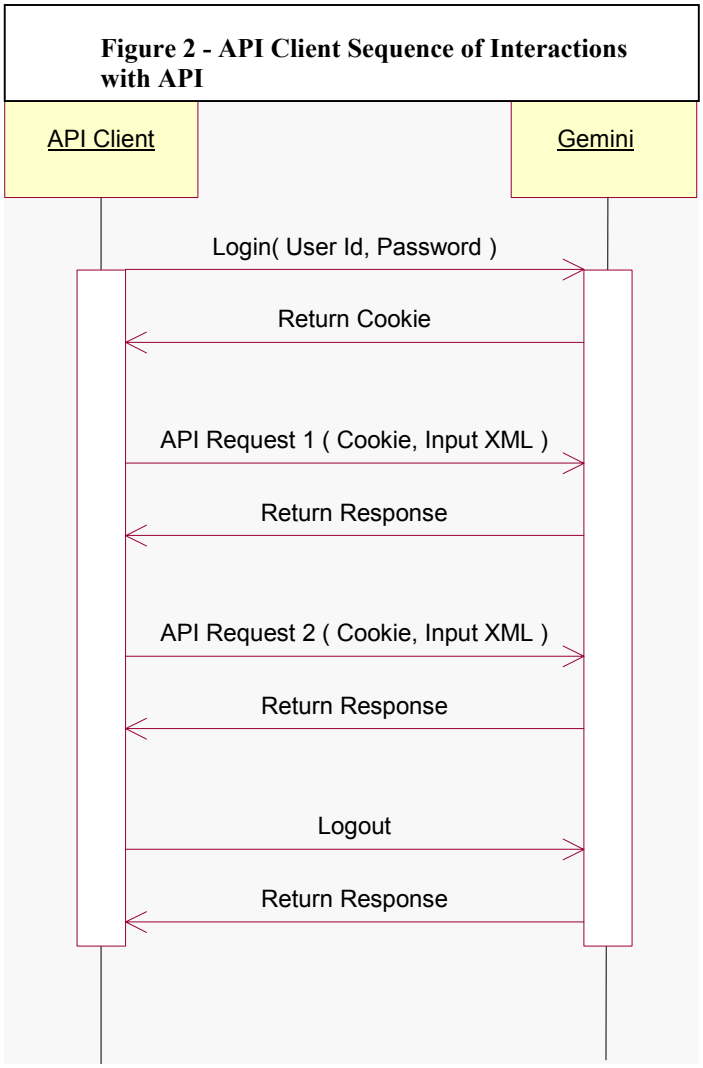
In particular, note that where we describe the HTTP interaction between API client and Gemini API in this guide, we are in fact describing the standard interaction between a

[1] Or, more precisely HTTPS, since data transfer between API clients and Gemini APIs is secured. However, we use the terms HTTP and HTTPS interchangeably without necessarily making the distinction.

web client (normally a web browser) and a web server, supported by the HTTP protocol. In this respect API client interaction with Gemini APIs differs only in that it is a two-way, XML based message exchange alternative to the serving of HTML to a web browser.

2 API Client Development Guidelines

The following diagram illustrates a typical sequence of interactions between an API client and Gemini during the lifetime of a session:



The following is the sequence of interactions illustrated above:

- API clients are authenticated using login credentials (user id and password). To trigger this, the API client invokes the login URL and provides a user id and associated password. These are passed through the HTTP headers.
- Gemini authenticates the user credentials supplied by the API client and, if successful, returns a session cookie as a response. A session cookie is sent to the API client along with each HTTP response. API clients **must** send the latest received cookie for the session with each subsequent request for an API.
- The API client constructs an XML document containing the input query parameters (view API) or data (update API). The XML document must conform to the corresponding request schema definition of the API being invoked. All API schemas are located on a National Grid server.

- The API client sends the XML document, along with the latest received cookies, to the URL specified for the required API.
- Gemini receives the request, parses the associated XML and services the requests. The response is sent as an XML document embedded in an HTTP message.
- The API client examines the HTTP response to check that the request was successful.
- If the request was successful, then the API client can extract the XML and interpret it as per the response schema definition of the API.
- If the request was **not** successful, then the API client must handle the reported error condition(s).
- One or more calls to the same or different APIs can be made in time sequence. The response (whether successful or not) to an API call **must** be received before submitting another API call within the same session.
- If the timeout period is exceeded between successive API calls then the session is timed out and the login step must be repeated to establish a new session before recommending API calls.
- The API client invokes the logout API to terminate the session when all desired requests/responses have been processed.
- Cookies must remain "within session", i.e., cookies returned by Gemini in one session must be not be sent to Gemini in another session.

There follows further clarification on a number of points relating to this interaction. Those points being:

- HTTPS and SSL
- Authentication and Authorisation
- Maintaining the Session
- Authentication/Authorisation Failures
- Request a Compressed Response to your API Client

2.1 *HTTPS and SSL*

APIs are accessed using the HTTPS protocol for secure data transfer. Gemini web servers have certificates from a valid Certification Authority. API clients must validate and retain the Gemini web server certificates. SSL is used for access.

To develop API clients any of the SSL toolkits such as OpenSSL, JSSE (provided by Sun Microsystems), etc., that support SSLv2, SSLv3 or TLSv1 may be used. All of these three SSL standards are supported.

2.2 *Authentication and Authorisation*

Gemini is a secure system. As such its URLs are protected[2]. To access a protected URL the user must have "Authenticated" them self and be "Authorised" to do so.

Authentication

This is the act of identifying yourself to Gemini by providing valid login credentials (user id and associated password).

Any request to a protected URL that does not supply a valid session cookie (see section 2.3 Maintaining the Session) will trigger authentication by Gemini. As such, it must include valid login credentials (see section 3.1 Login API) if it is to succeed.

Authorisation

This is the process of confirming that the user is permitted to invoke the functionality implied by the URL being accessed. Typically this is controlled via the roles assigned to the user.

It follows that authorisation must follow authentication, since it is necessary to identify the user before confirming their access rights.

2.3 *Maintaining the Session*

HTTP is a stateless protocol. In HTTP client/server interaction cookies are used to maintain a state/session. API client interaction with Gemini follows this approach.

If an API request to Gemini does not result in an authentication/authorisation failure, then Gemini returns session cookies in the API response. The next API request in that session must return the last received cookies.

It is essential that the last received cookies be returned. The API client must not, for example, attempt to send the cookies received at login if subsequent requests in that session have been processed. This will result in an authentication/authorisation failure, as the supplied session cookies will not be valid.

Remember also to remain synchronous within a session as advised in section 2 API Client Development Guidelines. Gemini maintains only one set of state information pertaining to a session. Therefore, if you submit asynchronous requests within a session you run the risk of corruption or loss of data.

2.4 *Authentication/Loss of Session/Authorisation Failures*

As explained previously, to invoke the functionality associated with any protected Gemini URL you must be authenticated and authorised. All Gemini functional APIs are invoked via protected URLs.

When accessing Gemini, failures can arise relating to the security infrastructure. These broadly fall into three categories:

- Authentication Failure;
- Loss of Session;

[2] The exceptions to this are the URLs associated with the change password screen and the change password API. Of course, these services still require valid login credentials to be supplied.

- Authorisation Failure.

The table, "Authentication/Loss of Session/Authorisation Failures" on pages 10 and 11 gives a detailed breakdown of possible causes of each of these failure categories. Each category is briefly explained immediately below (sections 2.4.1, 2.4.2 and 2.4.3).

2.4.1 Authentication Failure

Authentication of the supplied user credentials has failed. It follows that this scenario will **not** occur if your request supplied a session cookie, whether valid or not, because in that instance any supplied user credentials will be ignored. If your request did **not** supply a session cookie, then Gemini will demand that valid user credentials be supplied.

2.4.2 Loss of Session

This arises when there is a problem with the session cookie supplied by the API client. In this case the session is lost.

If user credentials were also supplied, then Gemini will attempt to establish a new session using those credentials. However, a request would not normally supply user credentials along with a session cookie. If it were believed that a valid session cookie is being supplied, then why would credentials be included also?

If a valid session cookie is supplied along with user credentials, then the user credentials are not checked.

2.4.3 Authorisation Failure

The request has been made within a valid, successfully authenticated session. However, the user is not authorised to access the specific functionality requested. This will be because the access to the requested API is not granted within the roles assigned to the calling user.

2.5 *Request a Compressed Response to your API Client*

Your API clients can request that the response returned to them be compressed. This will result in a reduction (typically 50% to 60%) in the data trafficked to your API client.

To activate this compression, the API client must send the following HTTP request header value.

Name of Header: Accept-Encoding
Value of Header: gzip

The client must also be able to uncompress the response on receipt.

If your client does not send this HTTP request header value, then the response it receives will not be compressed.

Table 1 - Authentication/Loss of Session/Authorisation Failures

Category	Cause(s)	Necessary Action	Further Remarks
Authentication Failure	Unknown User or Incorrect Password	Supply valid user credentials.	If you attempt to login in three times consecutively with an incorrect password then the account will become locked.
	Maximum Failed Number of Login Attempts Exceeded or User Disabled	Raise a help line call via your Local Security Officer (LSO) to get the account released.	A user account will be locked if the maximum number of consecutive failed login attempts (currently set to three) is reached. See "Unknown User or Incorrect Password". One further login attempt is then allowed every 30 minutes. A user can be disabled because the account has been inactive for longer than a set period, if this is configured. Alternatively, a SiteMinder administrator may have disabled the account for some reason.
	Change Password Forced, Password Expired, Immediate Password Change Required	Change the user's password before proceeding, either via the change password screen, or via the change password API.	There are some subtle distinctions between the causes. Change Password Forced Occurs if no password policies are in place and an administrator resets a password while dictating, "change on first use". Gemini is configured with password policies in place (e.g., expiry after thirty days, no reuse within twelve months, etc.). Consequently this cause should not occur. Password Expired Gemini is configured such that passwords will expire and must be changed after a set period of time. Password expiry will only take effect at the next login, i.e., if the password expires while you are in session then the session is not dropped. Immediate Password Change Required Similar to Change Password Forced, but this cause is expected to occur for Gemini as it corresponds to having password policies in place.

Loss of Session	Invalid Session, Revoked Session, Expired Session, Invalid Session Id	Repeat the login process to re-establish a valid session and proceed.	A session will expire if inactivity exceeds the session timeout period. This is currently configured to be after one hour of inactivity. Invalid session and invalid session identifier arise if the server cannot match the session cookie supplied with the request to a valid session. The most likely explanation is a bug in the API client's cookie handling.
Authorisation Failure	User not Authorised	Either contact IS Security via your LSO to get the role assignments of the userid changed or access the correct API.	If this scenario occurs then the userid is not authorised to access the particular API requested. There are two possible explanations: 1. The userid does not have appropriate roles assigned; 2. The API client is attempting to access the wrong API.

Further details on how your API client can identify occurrences of each of these causes of failure can be found in the table, "Using HTTP Response Headers to Identify the Cause of Authentication/Loss of Session/Authorisation Failures".

Several of the scenarios above make reference to policies in place in respect of passwords or sessions. At the time of writing, these policies are as detailed in the following table. This information is provided for guidance only and must not be considered binding or guaranteed. The same is true of any reference to these policies elsewhere in this document. These policies are configurable items and as such are subject to change as security policy, for example, dictates.

Table 2 - Password and Session Policies

Policy Type	Policy Sub-Type	Policy
Password	Expiration	- Passwords expire (password change forced) if not changed for 31 days. - Expiration warnings are issued for five days ahead of a password expiry. - Accounts are disabled after three successive attempts to login with an incorrect password. 30 minutes later one further attempt is allowed. If an incorrect password is given again, then 30 minutes later one further attempt is allowed, and so on. - A userid is disabled if not accessed for 90 days.
	Composition	- Passwords must be between six and eight characters in length. - No special characters, only lower and upper case letters and digits are allowed. - At least one digit must be included.

	Restriction	- Before a password can be reused a minimum of 365 days must have elapsed or at least five other passwords must have been used since that password was last used.
Session	Expiry	- A session expires after sixty minutes of inactivity.

3 Specifications for Session Management APIs

This section contains specifications for the session management APIs. These are the following APIs:

- Login
- Logout
- Change Password

These APIs are generic APIs that are needed to manage the sessions necessary to invoke of any of the functional APIs. The functional APIs are the APIs that actually invoke Gemini business logic. See section 5 for further detail on functional APIs.

Code Samples

Along with the specifications we provide code samples to illustrate the core, session management activities. These code samples are written using the Java programming language.

National Grid does not mandate the use of Java to write API clients. API clients must honour the mandated behaviour described in this document and the functional API specifications. The mandated behaviour is defined in terms of HTTP interaction and associated XML message exchange. Provided your API client conforms to this mandated behaviour, it can be constructed using any development technology.

Java has been used for illustrative code samples because:

1. It is in common use.
2. It does not abstract the underlying HTTP interaction so much as to mask the concepts that we are seeking to illustrate.

By contrast, Visual Basic API clients can be constructed that make use of the Microsoft provided XMLHTTPRequest object. This object encapsulates much of the default web browser behaviour that, as explained previously, must be mimicked by API clients. In doing so it masks that behaviour.

In summary, it is conceivable that the client interaction in your API clients may look very different from our Java samples, especially if you employ library objects that encapsulate much the default web browser behaviour. Do not be fooled into thinking that the mandated behaviour is not occurring and is not necessary.

NB. Sample code is provided for guidance only. It does not constitute code licensed or supported by National Grid. It is not covered by any warranty. National Grid does not provide or support API clients. It does support the API interface and its conformance to specification.

3.1 *Login API*

The login API can be used for the purposes of establishing a Gemini session prior to invoking functional APIs. The login API is simply a “dummy” protected URL. As it is a protected URL, when it is accessed the security infrastructure kicks in to authenticate and authorise the user. If authentication is successful, then a session is established.

Thus the login API allows a session to be established prior to making any request to a functional API. This is the login API's sole purpose.

It is also possible, if preferred, to establish a session by calling one of the functional APIs. The functional API URLs are also protected. If a request to any one of them does not contain a session cookie, then user credentials are checked and, if valid, a session will be established and the functionality of the API invoked in a single step.

The login API simply allows the separation of these two steps, establishing a session and invoking API functionality.

However you choose to establish your sessions, it is important that you continue to use open sessions where possible and do not unnecessarily continue to create new sessions by not reusing cookies corresponding to valid sessions. To do so would place an unnecessary burden on Gemini to the detriment of all users.

Note that generally Gemini userids authorised for screen access will not be authorised for API access and vice versa. You must be sure to request a userid that is authorised to access APIs for use by API clients.

URL to Access the API

API clients must invoke this URL to access this API's functionality:
/gemini/controllers/ApiLogin/

Note the trailing slash in this particular API URL. It is important that this is included.

3.1.1 HTTPS Request Headers

API clients can invoke the login URL for authentication. They must provide login credentials as a user id and password passed through HTTP headers.

Table 3 – Login API Request Headers

	Request Header Key	Value
1	Cookie	SMCHALLENGE=YES Required by the Netegrity SiteMinder software to invoke authentication.
2	Authorization	HTTP basic authentication is adopted for authenticating the user. The ID and the password must be concatenated together with a ":" delimiter in between, i.e., *userid:password*. The combined string must be encoded using Base64 encoding. The Base64 encoded value must be passed with this header. See http://www.ietf.org/rfc/rfc2617.txt for details of the Basic Access Authentication scheme and associated Base64 encoding.

Sample Code

```
/* Connect to Gemini */

url = new URL("https://<server>:<port>/<LoginURL>");
```

```
urlConnection=(HttpsURLConnection)url.openConnection();
urlConnection.setRequestMethod("POST");

/* Request Headers with ID and password sent to Gemini */

urlConnection.setRequestProperty("Cookie","SMCHALLENGE=YES");
String encodedLogin = base64Encode("userID", "password");
urlConnection.setRequestProperty("Authorization", encodedLogin);
```

3.1.2 HTTPS Response Headers

Case 1: Successful Authentication/Authorization

On successful authentication/authorization the following response will be sent to the API client.

Table 4 - Login API Response Headers – Successful Authentication/Authorization

	Response Header Key	Value
1	Set-Cookie	GEMINIAPIAUTHENTICATION=2001 GEMINIAPIAUTHENTICATION is used to indicate by Gemini to indicate the authentication outcome of any HTTPS request. 2001 indicates success and 4001 indicates failure.
2	Set-Cookie	GEMINIAPIAUTHORIZATION=2002 GEMINIAPIAUTHORIZATION is used to indicate by Gemini to indicate the authorization outcome of any HTTPS request. 2002 indicates success and 4002 indicates failure.
3	Set-Cookie	SMSESSION Cookie containing encrypted session ID. API clients must send the latest received cookies with every subsequent API invocation request.

Successful authentication/authorization via the login API will return an HTTP response 404 (file not found). 404 is returned on successful use of the login API because it is a "dummy" URL (see section 3.1 Login API) with no response page to be served. API clients should trap this response to ensure that it is not handled as an error.

By contrast, successful invocation of functional APIs (whether including user login or not) will return a HTTP response 200 (OK) as functional APIs do correspond to an underlying, functional URL that serves a response page.

Please be advised though, our guidance would be that you do **not** use HTTP response codes to infer the success or otherwise of your API calls. With this in mind we advise that API clients must be able to trap HTTP 400 and 500 series response codes and handle them gracefully, since they may not indicate an error. Despite this advice, we include some information on expected HTTP response codes in this guide for your information.

To diagnose errors you should instead rely on a combination of the following:

- GEMINIAPIAUTHENTICATION/GEMINIAPIAUTHORIZATION to determine the authentication/authorisation success or failure status.
- SMAUTHREASON (see below) to assist in determining the course of action necessary in the event of authentication/authorisation failures.
- XML success or error responses (see section 4 Error Handling) to detect and diagnose functionality errors.

Case 2: Authentication/Authorization Failed

In the case of authentication/authorisation failure the following response will be returned to the API client.

Table 5 - Authentication/Authorization Failed

	Response Header Key	Variable Value
1	Set-Cookie	GEMINIAPIAUTHENTICATION=4001
2	Set-Cookie	GEMINIAPIAUTHORIZATION=4002 Only for session related authentication/authorization failures. See Table 7 – Scenarios not Involving a Redirection.
3	LOCATION	SMAUTHREASON=<value> Can provide additional diagnostic information on authentication/authorisation failure depending on the exact failure scenario. See "SMAUTHREASON" below and Table 6 – Redirection Scenarios. **NB.** SMAUTHREASON is described here in relation to authentication/authorization failure, as it is usually associated with this scenario. However, there is at least one instance (password expiry warning) in which SMAUTHREASON provides information in relation to a **successful** authentication. This should be remembered.

SMAUTHREASON

In the event of an authentication/authorisation failure a redirect instruction is often returned in the response to the client. Under the HTTP protocol, such a redirection instruction is conveyed in the LOCATION header field.

Typically the redirect will be to the password maintenance service for user action. If Gemini screens are being used, then the web browser would follow this redirect instruction so that the user could take the necessary action, e.g., change of password.

It is not appropriate for an API client to follow that redirect to a screen based service. However, the redirect instruction contains important information as to the cause of the authentication/authorisation failure. This is encapsulated as a name/value pair parameter in the redirect URL. The parameter name is SMAUTHREASON.

Session related authorisation/authentication failures (invalid session, revoked session, expired session and invalid session identifier) and the invalid user credentials scenarios do **not** trigger a redirect instruction in the response to the client. In these situations a password maintenance action is not appropriate. Furthermore, since APIs uses basic

and not forms based authentication, there is no login form to redirect to for the purposes of re-establishing a session or correcting invalid user credentials.

In order to obtain supplementary information about the cause of failure, the API client must check for the LOCATION Response Header Key. The “SMAUTHREASON” name/value pair may contain supplementary information.

Sample Code

```
/* Response Header from Gemini */

boolean failureFlag = false;
int count = 0;
while(urlConnection.getHeaderFieldKey(count++) != null)
{
   String sKey = urlConnection.getHeaderFieldKey(count);
   String sValue = urlConnection.getHeaderField(count);
   if(sKey.equals("Set-Cookie"))
   {
      if(sValue.indexOf("GEMINIAPIAUTHENTICATION=2001")!=-1)
      {
         System.out.println("Successful authentication");
      }
      if(sValue.indexOf("GEMINIAPIAUTHORIZATION=2002")!=-1)
      {
         System.out.println("Successful authorisation");
      }
      if(sValue.indexOf("GEMINIAPIAUTHENTICATION=4001")!=-1)
      {
         System.out.println("Authentication failed");
         failureFlag = true;
      }
      if(sValue.indexOf("GEMINIAPIAUTHORIZATION=4002")!=-1)
      {
         System.out.println("Authorisation failed");
         failureFlag = true;
      }
      /* In case of successful authentication/authorisation store the
         session cookie */
      if(!failureFlag && sValue.indexOf("SMSESSION")!=-1)
      {
         setLatestSessionCookie(sValue);
      }
   }
   /* LOCATION header key needs to be checked only in case of
      authentication/authorisation failure to obtain supplementary
      information */
   if(failureFlag && sKey.equalsIgnoreCase("LOCATION"))
   {
      if(sValue.indexOf("SMAUTHREASON=1")!=-1)
      {
      System.out.println("User must change password");
      }
   }
```

```
   /* Similar checks must be done for other SMAUTHREASON values */}
}
```

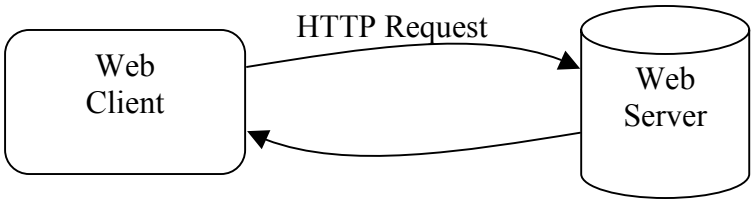
3.1.3 HTTP Redirection

HTTP redirection is characterised by one of the 3xx series of HTTP status codes. An explanation of redirection 3xx status codes, which includes a brief description of HTTP redirection, can be found at the W3C web site. The specific redirection status code that you will encounter in your interaction with Gemini is the 302 Found status code.

To accurately diagnose the causes of authentication/authorization failure in your API clients you **must** understand HTTP redirection. Furthermore you **must** use a technology that allows you to trap and interpret the "intermediate" (see below) response in a HTTP redirection.

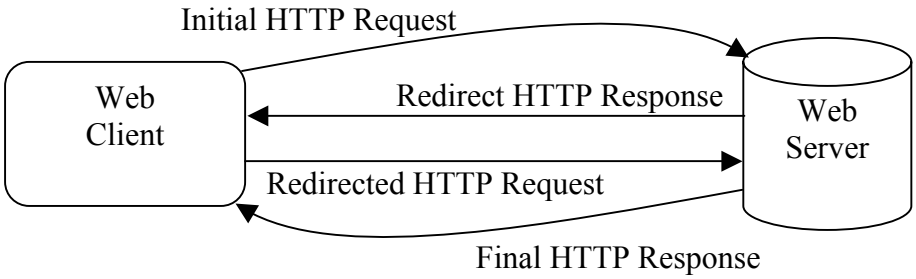
In a web client/server interaction that doesn't involve a redirection 3xx status code there is one request/response pair. The web client issues a HTTP request and receives a HTTP response and at that point the interaction ends.

This is illustrated in the following diagram:



In a web client/server interaction that does involve a redirection 3xx status code there are two (at least) request/response pairs. The web client issues an initial HTTP request and receives a "follow me" redirect HTTP response. This redirect response instructs the web client to issue a redirected HTTP request to a location that it specifies. When the web client follows this instruction via a redirected HTTP request, the web server issues the final HTTP response and the interaction ends.

This is illustrated in the following diagram:



It should be noted that the behaviour described is mandated for web clients/servers that implement the HTTP protocol as specified. In this respect the behaviour expected of your API client is no different to the behaviour expected, and honoured, by any web browser. Web browsers and API clients are merely sub-classes of web clients expected to conform to the HTTP standards expectations of a web client.

The Redirect HTTP Response

The redirect HTTP response is characterised by having a 302-status code. It also contains the URI that the web client is instructed to go to via a redirected HTTP request. This URI is specified in the LOCATION header field of the redirect HTTP response.

An explanation of the LOCATION header field can again be found at the W3C web site. Again, none of this is specific to Gemini.

API Client Considerations

Most objects simulating web client behaviour will by default automatically handle redirect requests and only return the final response. Therefore, if you want to trap and interpret the content of a redirect HTTP response, then your choice of development tools for your API clients must allow you to do this. To illustrate this we discuss two real life examples.

There are of course many different technologies that you can employ to write API clients. We cannot have knowledge of them all. However, we provide these specific examples to illustrate the general principles of redirection that you must consider.

Use of the MSXML2.ServerXMLHTTP40 Object in Visual Basic

API developers using the MSXML2.ServerXMLHTTP40 object within a Visual Basic API client have on occasion contacted us. They report that they cannot intercept and interpret the redirect response.

Our investigations have revealed that this is a known problem with this object. See these Internet forum postings:

http://support.microsoft.com/default.aspx?scid=kb;EN-US;q308607

http://www.eggheadcafe.com/ng/microsoft.public.xml.msxml-webrelease/post20427556.asp

The advice in both postings is that if you want to intercept redirects you should use the WinHttpRequest object instead.

This is an example of an object that implements the standard behaviours of a web client and exposes them for a wrap around language to access. One such standard behaviour is the following of a redirect. The point illustrated is that if the object you use does not allow the wrap around language to trap and interpret the redirect, then this behaviour is not exposed.

Use of the HttpsURLConnection Object in Java

Similarly the HttpsURLConnection object in Java will by default follow redirects to the final response and deny the Java code sight of the redirect response and redirected request.

You can change this behaviour using the setInstanceFollowRedirects method as follows:

HttpsURLConnection httpsConn = (HttpsURLConnection)url.openConnection();

httpsConn.setInstanceFollowRedirects(false);

Where url is of the URL object type.

Gemini Redirection Scenarios

Here we describe the Gemini Redirection Scenarios that you will encounter so that you can look for them and handle them as you wish in your API clients. This section is deliberately **not** called Gemini **API** Redirection Scenarios, as these redirections are not specific to APIs. They are encountered when accessing the screens also but silently handled by your web browser without your knowledge.

Firstly, in all of these redirection scenarios there will only be a single redirection before the final response is reached. The HTTP protocol allows for multiple redirections.

The scenarios in which you can expect to receive a redirection are as follows:

1. Any authentication/authorisation failure scenario that provides further information via an SMAUTHREASON code.
2. The password expiry warning response. In this instance there hasn't been an authentication/authorisation failure but warning that password expiry is imminent is notified via an SMAUTHREASON code in the LOCATION header field.
3. The logout API issues a redirect request when used to terminate a session.

The complete behaviour in these scenarios is detailed in the table "Redirection Scenarios" on the next page.

Table 6 - Redirection Scenarios

	HTTP Response				
	Redirect				Final
	HTTP Status	SMAUTHREASON	GEMINIAPIAUTHENTICATION	GEMINIAPIAUTHORIZATION	HTTP Status
Authentication / Authorisation Failure					
Maximum Failed Number of Login Attempts Exceeded	302	24[3]	4001	*None*	200
User Disabled (by Administrator)		7			
User Disabled (Due to Inactivity)		25			
Change Password Forced		1			
Password Expired		19			
Immediate Password Change Required		20			
Password Expiry Warning[4]		18	2001		
Logout API		0	*None*		

Notes:

- GEMINIAPIAUTHENTICATION header field is **not** present in the final response following a redirection.
- Any SMAUTHREASON value returned present in the LOCATION header of the redirect response will then appear in the URL query string of the final response. This is as a direct consequence of following the redirect.
- In the case of a Password Expiry Warning, if you automatically follow the redirect to the final URL then no session cookies will be returned. As a consequence you will not be able to call functional APIs. If you wish to ignore the password expiry warning and change your password at a later time then you must intercept the redirect response in order to use the session cookies provided.

Though it isn't directly related to the topic of redirection, for completeness we record the results of the authentication/authorisation failure scenarios that do **not** involve redirection in the table "Scenarios not Involving a Redirection" on the next page.

[3] This will appear on the third consecutive login attempt with an incorrect password. One further attempt is then allowed every half hour. This too will return SMAUTHREASON=24 if the password is again incorrect.
[4] Although it is detailed in this table, this is not actually an authentication/loss of session/authorisation failure scenario. A password expiry warning informs the client that the password is due to expire shortly. However, authentication has been successful.

Table 7 - Scenarios not Involving a Redirection

	HTTP Response		
	HTTP Status	GEMINIAPIAUTHENTICATION	GEMINIAPIAUTHORIZATION
Authentication/Authorisation			
Unknown User	401	4001	*None*
Incorrect Password			
User not Authorised		*None*	
Session			
Invalid Session	401	4001	4002
Revoked Session			
Expired Session			
Invalid Session Id			

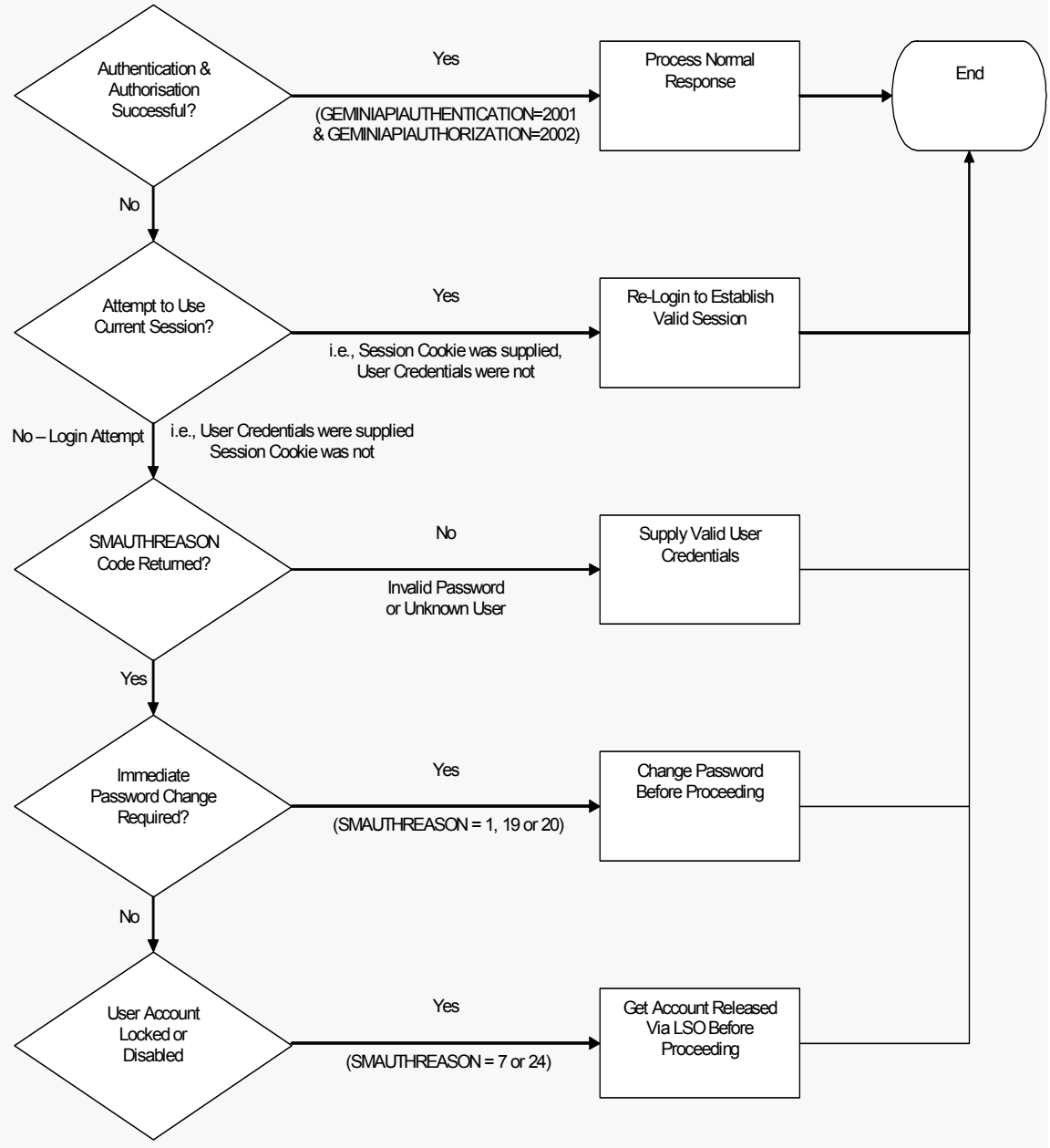
Note that loss of session failures would not be expected to occur when accessing the login API, since the sole purpose for accessing the login API is to establish a session. In this instance no session cookies associated with an existing session would be supplied. Session loss would normally be associated with functional API calls within the session. However, we present the full list of response codes in one place, with the login API specification.

An illustration of how logic might be constructed to determine the cause of authentication/loss of session/authorisation failures is given on the next page. Remember, this diagram is provided to illustrate a **possible** not mandated or recommended approach.

3.1.4 HTTPS Response Body

The HTTPS response body returned by the login API is null.

Figure 3 - Example Logic for Determining the Cause of Authentication/Loss of Session/Authorisation Failures



3.2 *Functional APIs*

The basic principles of invoking functional APIs are described here. The specifications of each of the functional APIs reside in separate documents; see section 5 Functional API Specifications.

3.2.1 HTTPS Request Headers

The cookies that were last received by the API client must be passed via HTTPS headers on the next API call. API clients must use the POST method to submit their requests.

Note: When invoking functional APIs you must **not** set a value of "text/xml" for the Content-Type header key. This can result in GEM_API_ERROR_0001 ("XML document is not valid") messages returned for perfectly valid XML.

Table 8 - Functional API Request Headers

	Request Header Key	Value
1	Cookie	All the latest received cookies provided by Gemini web servers must be returned with every subsequent API invocation request.

Sample Code

```
/* Connect to Gemini */

url = new URL("https://<server>:<port>/<API-URL>");
urlConnection=(HttpsURLConnection)url.openConnection();
urlConnection.setRequestMethod("POST");
int count = 0;
while(urlConnection.getHeaderFieldKey(count++) != null)
{
   String sKey = urlConnection.getHeaderFieldKey(count);
   String sValue = urlConnection.getHeaderField(count);

   if(sKey.equals("Set-Cookie"))
   {
      urlConnection.setRequestProperty("Cookie", sValue);
   }
}
```

3.2.2 HTTPS Request Body

The API client must send an XML document that contains the input parameters required by the specific API being called[5]. This is passed via the INPUT name/value pair in the HTTPS request body. For a single API request only one name/value pair can be sent.

[5] There is one exception to this principle and that is the System Status Information API. This API requires no input data since it simply returns the latest system status information.

Table 9 - Functional API Request Body

	Name	Value
1	INPUT	Valid XML input document that contains input parameters sent to the server for processing in response to the request.

Sample Code

```
/* Request body sent to Gemini */

String strXMLParams = "INPUT="+xmlDocument.toString();
outputStream.write(strXMLParams.getBytes());
outputStream.flush();
```

3.2.3 HTTPS Response Headers

Table 10 - Functional API Response Headers

	Response Header Key	Value
1	Set-Cookie	SMSESSION and other values (All latest received cookies provided by Gemini web servers must be returned with every subsequent API invocation request. It isn't necessary to understand or interpret the Set-Cookie content, just so long as the general principle of echoing it back is followed).

Sample Code

```
/* Response header from Gemini */

int count = 0;
while(urlConnection.getHeaderFieldKey(count++) != null)
{
   String sKey = urlConnection.getHeaderFieldKey(count);
   String sValue = urlConnection.getHeaderField(count);

   if(sKey.equals("Set-Cookie"))
   {
      /*Retrieve All the Cookies send as response.
      These cookies must be returned with every subsequent API
      invocation request. */
   }
}
```

3.2.4 HTTPS Response Body

A valid XML document will be sent as the response body. This XML will conform to the output schema definition of the corresponding API. Any failure to do so should be reported to National Grid as an error.

3.3 *Logout API*

The logout API is a "session killer". If it is invoked with a valid session cookie, then that session is logged out. It is good practice that when you have finished using a session you invoke the logout API. Inactive sessions will eventually be time expired but using the logout API when you are finished with a session helps to manage the number of open sessions.

URL to Access the API

API clients must invoke this URL to access this API's functionality:
/home/common/jsp/smlogout.jsp

3.3.1 HTTPS Request Headers

	Request Header Key	Value
1	Cookie	All the latest received cookies provided by the Gemini web servers must be returned with the request to the logout API.

When successful (i.e., session has been logged out), the logout API return a 302 HTTP response code.

3.4 *Change Password API*

This API permits the Gemini application password changes. User Id, Old Password and New Password are required as input parameters of the HTTP Post request. On execution of this API, the API Client will receive an HTTP response with either the confirmation of the password change, or an error message. The implementation of this API differs from the other Gemini functionality API's (presented in this document) as it invokes SiteMinder DMS API's for the password change.

URL to Access the API

API clients must invoke this URL to access this API's functionality:
/gemini/controllers/ChangePasswordControllerAPI/

3.4.1 HTTP Request Headers

No headers identified.

3.4.2 HTTP Request Body

The API Client passes the following input parameters as the HTTP request body.

Table 11 - Change Password API Request Parameters

Name	Value
USER_ID	String containing User Id.
OLD_PASSWORD	String containing old password.
NEW_PASSWORD	String containing new password.
CONFIRM_NEW_PASSWORD	String containing new password.

3.4.3 Sample Request Code

Sample Request coded in Java:

```
/* Connect to Gemini */
url = new
URL("https://<server>:<port>/gemini/controllers/ChangePasswordControllerAPI");
urlConnection = (HttpsURLConnection)url.openConnection();
urlConnection.setRequestMethod("POST");

/* Declare and set the string variables USER_ID, OLD_PASSWORD, NEW_PASSWORD
and CONFIRM_NEW_PASSWORD with the appropriate values */

/* Request body sent to Gemini */
OutputStream outputStream  = urlConnection.getOutputStream();
outputStream.write( ("USER_ID=" + USER_ID + "&").getBytes());
outputStream.write( ("OLD_PASSWORD=" + OLD_PASSWORD + "&").getBytes());
outputStream.write( ("NEW_PASSWORD=" + NEW_PASSWORD + "&").getBytes());
outputStream.write( ("CONFIRM_NEW_PASSWORD=" +
CONFIRM_NEW_PASSWORD).getBytes() );
outputStream.flush();
```

3.4.4 HTTP Response Body

The following response codes will be returned as part of the HTTP response body.

Table 12 - Change Password API Response Codes

Response Code	Message
GEM_API_SEC_1000	Your new password has been set. Use this new password the next time you log into your account
GEM_API_SEC_ERR_1001	Your password change was not accepted. Please try again.
GEM_API_SEC_ERR_1002	Please match your new password and confirmation.
GEM_API_SEC_ERR_1003	Access is restricted to authorised users only.
GEM_API_SEC_ERR_1004	System encountered an error. Please try again after some time.
GEM_API_SEC_ERR_1005	You cannot access your account because you have exceeded the limit of login attempts. Please contact your Security Administrator or Help Desk.

3.4.5 Sample Response Code

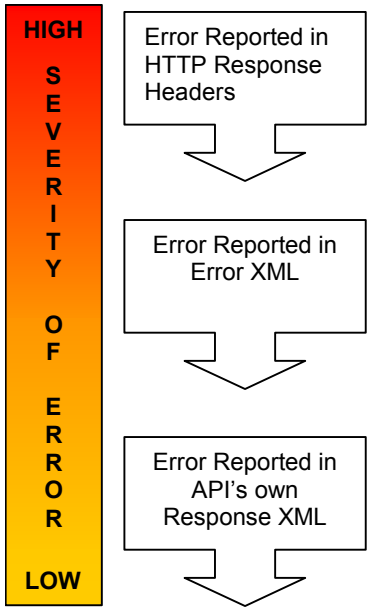
```
/* Process response code in API Client program */

/* It is assumed the URL has been requested. Please refer to the earlier code
snippet */

InputStream is = urlConnection.getInputStream();
InputStreamReader ins = new InputStreamReader(is);
BufferedReader br = new BufferedReader( ins );
```

```
/* To read the response code */
String CODE = br.readLine();

/* To read the message */
String MESSAGE = br.readLine();

If ( CODE.equals(GEM_API_SEC_1000) )
{
    System.out.println("Password Changed successfully");
    ...
    ...
    ...
}

is.close();
ins.close();
br.close();

urlConnection.disconnect();
```

4 Error Handling

It is useful to consider error handling by Gemini APIs in terms of a hierarchy, with the most severe errors at the top and the least severe at the bottom. This is illustrated in the following diagram:

Figure 1 - Error Handling by Severity of Error



Errors reported via the HTTP response headers will generally relate to authentication/authorisation failures. See sections 2.4 Authentication/Loss of Session/Authorisation Failures and 3.1.2 HTTPS Response Headers for further information on how to recognise and handle these errors.

Assuming that an error of this severity has not occurred, any errors will be reported via XML. API clients can check the HTTPS response header "content-type" for the value "text/xml" to determine if the response from the API contains an XML document. If the response does not contain an XML document, then this header value will be set to something other than "text/xml".

Note that errors reported via XML are not relevant to the session management APIs (see section 3 Specifications for Session Management APIs) as these APIs do not exchange XML with the API client.

The rest of this Error Handling section concerns errors reported via XML.

4.1.1 Errors Reported via XML

Generally, when the processing of a request from an API Client fails, an XML response is returned containing the error message in an Error XML structure. This Error XML structure is defined in sections 4.1.1 and 4.1.2. There is however an exception to this rule for APIs that add or modify Gemini data.

Update APIs process one record at a time, and a success or failure message can be returned for each record in the specific API's own response XML. This is explained further through the scenarios detailed below.

Query APIs

For APIs that query data from Gemini, API Clients can expect one of the following 3 scenarios:

Table 13 - Error Scenarios for Query APIs

Scenario		Response XML	Error XML
a)	Successful Query – A successful transaction that returns the query results in the API's response XML as defined in the relevant API specification under ''API Response" for that API.	Yes	No
b)	Generic Error – A generic or system error (e.g. Invalid XML) that prevents the transaction from executing successfully. No query results are returned. An Error XML containing the appropriate error code is generated and returned to the API Client.	No	Yes Error Codes in Section 4.1.4
c)	API Specific Error – An API specific error (e.g. Invalid Meter Id for the BA) that prevents the transaction from executing successfully. No query results are returned. An Error XML containing the appropriate error code is generated and returned to the API Client.	No	Yes Error Codes in Individual API Specifictions

Update APIs

For APIs that add or modify Gemini data (also known as Update APIs), one of the following scenarios can occur:

Table 14 – Error Scenarios for Update APIs

Scenario		Response XML	Error XML
a)	Fully Successful Update – A transaction in which **all** input records were successfully added or updated. The API Client receives a response XML as defined in the relevant API specification under "API Response" for that API. Against each input record, this XML contains Message Code (MSG_CD) and Message Description (MSG_DESC) indicating that the record was updated successfully. Response codes for these messages are of the format **GEM_API_MSG_nnnn**.	Yes Response Codes in Individual API Specifications	No

b)	Partially Successful Update – A transaction in which **some** input records were successfully added or updated, while others were erroneous. The API Client receives a response XML as defined in the relevant API specification under "API Response" for that API. Against each input record, the status for that record is provided through the Message Code (MSG_CD) and Message Description (MSG_DESC) elements. Records that were updated successfully have response codes of the format **GEM_API_MSG_nnnn**, while records that were not updated contain error codes of the format **GEM_API_ERROR_nnnn**, along with a message describing the error.	Yes Response Codes in Individual API Specifications	No
c)	Unsuccessful Update – A transaction in which **none** of the input records were successfully added or updated. The API has processed each input record, but none of the records were successful. The API Client receives a response XML as defined in the relevant API specification under "API Response" for that API. Against each input record, the error for that record is indicated through the Message Code (MSG_CD) and Message Description (MSG_DESC) elements. Error codes are of the format **GEM_API_ERROR_nnnn**.	Yes Response Codes in Individual API Specifications	No
d)	Generic Error – A generic or system error (e.g. Invalid XML) which prevents the transaction from executing successfully. The API has not processed any input records. No records are updated. An Error XML containing the appropriate error code is generated and returned to the API Client.	No	Yes Error Codes in Section 4.1.4

4.1.2 Error XML Specification

Hierar-chy	Data Element	Description	Data Type	Data Length	Mandatory
0	Errors	Top-level hierarchy for errors.			
1	errInfo	Top-level hierarchy for error elements. One or many errors may be returned. Attribute 'ID' of this element, shows the sequence identifier of the error.			
2	errCode	Error Code	String	18	Y
2	errDesc	Error Description	String	400	Y

4.1.3 Error Schema Definition

URL of file: **/gemini/api/schema/geminiapierror.xsd**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="errors">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="errInfo" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="errCode" type="xs:string"/>
```

```
                    <xs:element name="errDesc" type="xs:string"/>
                  </xs:sequence>
                  <xs:attribute name="ID" type="xs:int" use="optional"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
</xs:schema>
```

4.1.4 Generic Error Codes

The table below lists error codes that may be returned by a call to any API (Query or Update). These errors are returned in the Error XML specified in sections 4.1.2 and 4.1.3. Please note that although error codes will not change, error message text is subject to revision without formal notice.

Table 15 - Generic API Error Codes

Error Code	Error Message
GEM_API_ERROR_0000	System error
GEM_API_ERROR_0001	XML document is not valid[6]
GEM_API_ERROR_0002	No record(s) found
GEM_API_ERROR_0003	The record could not be saved
GEM_API_ERROR_0007	Request not serviced. Usage limit exceeded.

[6] BAs may use XML schema definitions published by National Grid to validate input and output XML documents. This is particularly important for your input XML. You are responsible for ensuring that this conforms to the XSD specification.

5 Functional API Specifications

Gemini API specifications for functional APIs are contained in separate documents, one for each API. The sections contained within these documents are as follows:

- Overview
- API URL
- XSD URL
- Request XML Specification
- Response XML Specification
- API Specific Error Messages

Overview

Provides a brief description of the function of the API.

API URL

The URL that is the API can be called at.

All URLs (both API and XSD) are given relative to the server root path for the API service.

Absolute URLs are not given. These will reflect, for example, National Grid's latest domain policy and are particularly susceptible to change. They are therefore communicated separately from the UK-Link controlled documentation set.

API client developers are advised to parameterise URLs.

XSD URL

The URLs of the request and response XSD associated with the API.

Request and response XSDs are made available on our web servers for API developers to use to validate the format of their XML input documents[7]. This is separate from and can be used in addition to the error messages returned by APIs, see section 4 – Error Handling.

Request XML Specification

Defines the XML structure that the requests to the API must conform to. This is shown both as actual XSDs and in a more descriptive tabular format.

Note that Data Type columns in the tables are general descriptions for the benefit of the reader. For exact XML data types please refer to the corresponding Schema Definition.

Unless specified otherwise the values for the XML fields identified in these documents should **not** be padded.

Response XML Specification

As for Request XML Specification but corresponding to the response XML document rather than the request XML document.

[7] The error XSD is also available on our web servers. See section 4.1.3 Error Schema Definition in this document. Of course, the error XSD is not specific to particular functional APIs.

In some instances a data length is not specified in the descriptive tables for response data elements. This is where the length is not known in advance, e.g., dependent on calculation or not directly taken from a database record, etc., and so the upper limit is not known in advance.

Error Handling

Contains details of error messages specific to the API in question, including details of where returned error messages would appear. Not that for update APIs success messages, indicating a successful update, are also defined.

5.1 *API Scope*

Note that the implementation date of different APIs may be vary. You should refer to the individual API specification documents to understand when that API is available from.

5.1.1 Entry Capacity

Table 16 - Entry Capacity Functional APIs

API Name	Equivalent Gemini Online Screen	Accessible By
All Active Bids		Shipper
Bid Information Shipper View – Normal Pricing Strategy		Shipper
Capacity Trade Details		Shipper
Capacity Trade Registration		Shipper
Daily Auctions Summary Report – Bids / Shippers	Product → Publish Reports → MoS Results → Bids / Shippers	Shipper
Daily Auctions Summary Report – Max / Min Price Information	Product → Publish Reports → MoS Results → Max / Min Price Info	Shipper
Daily Auctions Summary Report – Sold / Unsold	Product → Publish Reports → MoS Results → Sold / Unsold	Shipper
Daily Auctions Summary Report - WAP	Product → Publish Reports → MoS Results → WAP	Shipper
Revised Entitlements	Product → Entitlements-NET	Shipper

5.1.2 Energy Balancing

Table 17 - Energy Balancing Functional APIs

API Name	Equivalent Gemini Online Screen[8]	Accessible By
Add / Update Nominations	Nominations → Nominations → (Update)	Shipper
Confirm Multi Locational Trades (Update)	OCM → Maintain OCM Renominations → (Confirm Multi Loc)	Shipper
Daily Cashout Tolerance Breakdown	Allocations → Cashout → Cashout Tolerance Breakdown	Shipper
Gas Trades Breakdown	OCM → Trade Details → OCM / OTC	Shipper / Market

[8] Options shown in brackets relate to screen buttons

API Name	Equivalent Gemini Online Screen[8]	Accessible By
	Trade Breakdown Summary	Operator
Maintain OCM Renominations (View)	OCM → Maintain OCM Renominations	Shipper
Maintain Physical Renominations (Update)	OCM → Maintain OCM Renominations → (Confirm NBP Phy)	Shipper
Meter to Zone Relationship	Meter Details → Setup Meter Details → Maintain Relationship	Shipper / Claims Validation Agents
Price Information History	OCM → Price Information History	Shipper / Market Operator
Register Physical/Locational Trades	OCM → Physical Locational Trades	Market Operator
Register Title Swaps	OCM → Title Swap Trades → Register Title Swaps	Market Operator
Shipper EOD Noms (Hour Bar)	Nominations → INS → BA EOD/INS Imbalance	Shipper
Shipper Preliminary Balance	Nominations → Balance → Business Associate Balance	Shipper / Market Operator
Shipper Total Energy Forecast	Nominations → Demand Attribution → BA Total Energy Forecast	Shipper
System Nomination Balance	Nominations → Balance → System Balance	Shipper
System Status History	Nominations → Balance → System Status Information - History	Shipper
System Status Information	Nominations → Balance → System Status Information	Shipper
Update INS Nominations	Nominations → INS → Nominations → (Add or Modify)	Shipper
Update Renominations	Nominations → Renominations (Update)	Shipper
View INS Nominations	Nominations → INS → Nominations	Shipper
View Renomination Details	Nominations → Renominations → (V)	Shipper
View Renominations	Nominations → Renominations	Shipper
View Shipper Trade Details	OCM → View OCM Trade Details → View Shipper Trade Details	Shipper
View Storage Output Claims	Allocations → Pre Closeout Claims → Storage Output Claims	Claims Validation Agents
View WCF/SF Values	Allocations → LDZ → Demand Attribution → WCF / SF	Shipper

5.2 *Date Formats*

Gemini API schemas use the standard data type formats defined by W3C. All date fields use the **CCYY-MM-DD** format. All timestamp values use the **CCYY-MM-DDThh:mm:ss** format.

- **C** represents a digit used in the thousands and hundreds components, the "century" component of the time element "year". Legal values are from 0 to 9.
- **Y** represents a digit used in the tens and units components of the time element "year". Legal values are from 0 to 9.
- **M** represents a digit used in the time element "month". The two digits in an MM format can have values from 1 to 12.

- **D** represents a digit used in the time element "day".
  - The two digits in a DD format can have values from 1 to 28 if the month value equals 2
  - 1 to 29 if the month value equals 2 and the year is a leap year
  - 1 to 30 if the month value equals 4, 6, 9 or 11 and
  - 1 to 31 if the month value equals 1, 3, 5, 7, 8, 10 or 12.
- **h** represents a digit used in the time element "hour". The two digits in a hh format can have values from 0 to 23.
- **m** represents a digit used in the time element "minute". The two digits in an mm format can have values from 0 to 59.

s represents a digit used in the time element "second". The two digits in an ss format can have values from 0 to 59.

6 Element Name Abbreviations

The following abbreviations for API element names are used throughout the API specifications. We include tables giving you the full names here for your convenience.

6.1 *Entry Capacity APIs*

Shortened Element Name	Element Description
ACTV_BIDS_OUT	All Active Bids Output
ALL_ACTV_BIDS_QRY	All Active Bids Query
BID_CPCTY	Bid Capacity
BID_INFO_NRML_PRCNG_OUT	Bid Information Normal Pricing Output
BID_INFO_NRML_PRCNG_QRY	Bid Information Normal Pricing Query
BID_PRC	Bid Price
BID_STS	Bid Status
BID_TMSTMP	Bid Time Stamp
BID_TYP	Bid Type
BID_VAL	Bid Value
BIDWND_END_DT	Bid Window End Date
BIDWND_STRT_DT	Bid Window Start Date
BUY_BA	Buying BA
CNTRCT_TYP	Contract Type
CPCTY_DAY	Capacity Day
CPCTY_TRD_DTL_OUT	Capacity Trade Detail Output
CPCTY_TRD_DTL_QRY	Capacity Trade Details Query
CPCTY_TRD_REGN_INP	Capacity Trade Registration Input
CPCTY_TRD_REGN_OUT	Capacity Trade Registration Output
ENTLMNT	Entitlement
ENTLMNT_PRC	Entitlement Price
ENTLMNTS_OUT	Entitlements Output
ENTLMNTS_QRY	Entitlements Query
ERLST_EXRCSBL_DAY	Earliest Exercisable Day
ERLST_EXRCSBL_TM	Earliest Exercisable Time
EXER_PRC	Exercise Price
FLOW_STRT_TM	Flow Start Time
LCTN	Location
LCTNS	Locations
LTST_EXRCSBL_DAY	Latest Exercisable Day
LTST_EXRCSBL_TM	Latest Exercisable Time
MAX_QTY	Maximum Quantity
MIN_QTY	Minimum Quantity
MOS	Method Of Sale
NET_CPCTY	Net Capacity
NET_FRM_CPCTY	Net Firm Capacity
NET_INTR_CPCTY	Net Interruptible Capacity
NET_SCLD_INTR_CPCTY	Net Scaled Interruptible Capacity
OPTN_BID	Option Bid
PRD_FRM	Period From
PRD_TO	Period To
PRDT	Product
PRDT_CTGRY	Product Category
PRDT_TYP	Product Type
PRM_PRC	Premium Price
QRY_CRTR_1	Query Criteria 1

QRY_CRTR_2	Query Criteria 2
QRY_CRTR_3	Query Criteria 3
RMNG_EXRCSBL_DAYS	Remaining Exercisable Days
RSN_FOR_RJCTN	Reason For Rejection
SELL_BA	Selling BA
SRVC_TYP	Service Type
SUBTNDR_ID	Sub Tender ID
SUBTX_END_DT	Sub Transaction End Date
SUBTX_PRD_FRM	Sub Transaction Period From
SUBTX_PRD_TO	Sub Transaction Period To
SUBTX_STRT_DT	Sub Transaction Start Date
TMSTMP	Time Stamp
TRD_PRC	Trade Price
TRD_QTY	Trade Quantity
TRD_REF_NMBR	Trade Reference Number
TRD_STS	Trade Status
TRD_TM	Trade Time
TRNCH_NMBR	Trance Number
TX_END_DT	Transaction End Date
TX_STRT_DT	Transaction Start Date

6.2 *Energy Balancing APIs*

Shortened Element Name	Element Description
ACTV_IND	Active indicator
ACTVTY	Top level hierarchy for activity elements
ACTVTY_NBR	Activity number
AGNT	Agent for whom the storage output claims are retrieved
AGNT_NM	Indicates the agent name.
ALLCTD_FCAST	Allocated forecast.
ALLOC_CPCTY	Allocated Bid Capacity
AMT_BGT	Amount Bought
AMT_SALE	Sale Amount
BA_CD	The BA three character short code
BID_CPCTY	Bid Capacity
BID_ID	Bid Identifier
BID_ORGNTOR	Unique identification code for a BA.
BID_STS	Bid Status
BID_TMSTMP	Bid Timestamp
BID_TYPE	The field indicating whether the bid is a locational or a physical bid
BIDWND_END_DT	Bid Window End Date
BIDWND_STRT_DT	Bid Window Start Date
BTCH_ID	Identification for Transco physical/locational Trades
BUY_DTL	Top-level hierarchy for buy details.
BUY_SELL	Buy Sell Indicator
BUY_SELL_IND	Buy Sell Indicator
CLM_DTL	Top-level claim details.
CLMD_QTY	Indicates the claimed quantity
CLNDR_DAY	Calendar Date
CLRNG_PRC	Clearing Price
CMT_TMSTMP	Commit timestamp
CNG_SNCE_LST_NM	The percentage change since the last NDMA Forecast.

CNTRCT_TYP	Contract Type
CRT_DT	Creation date
CRT_TM	Creation time
CSHOUT_BRKDN_DTL	Top-level hierarchy for cashout breakdown details.
CSHOUT_BRKDN_OUT	Top level hierarchy for cashout tolerance breakdown output elements
CSHOUT_BRKDN_QRY	Top level hierarchy for cashout tolerance breakdown query elements
CSHOUT_TLRNCE	BA's cashout tolerance quantity
CST	Cost
CURR_DMND	The current forecast for the EOD demand (Mcm)
CV	Calorific value
DAY_RATE	Day Rate
DEEMED_ALLOC_NDMA_LOAD	Deemed Allocated NDMA Load Quantity for the BA, for the gas day
DLY_CSHOUT_BRKDN_DTL	Top-level hierarchy for daily cashout tolerance breakdown details.
DM_SHR	Sum of all the DMC DMA and Shrinkage Nominations for a BA
DMC_TYPE	The sub type for a Daily Consumer (DC) meter.
DTL	Detail (a top level hierarchy for header fields).
EFF_END_DT	Effective end date
EFF_ST_DT	Effective start date
END_ACTVTY_NBR	End activity number
ENRG_DTL	Top level hierarchy for energy detail
ENRGY_TOTLS	Top-level hierarchy for the energy totals.
EXER_PRC	Exercise Price
FCAST_DEVTN	BA's forecast deviation for a gas day
FIRM_METER_NO	The firm meter number for the storage output meter.
FLEXBLTY	Sum of all flexibility trades (Buy or Sell) for a BA
FLOW_TYPE	Indicates if the Flow Type is firm or interruptible
FRST_NM	First nomination
GAS_DAY	Gas day
GAS_TRD_BRKDN_OUT	Top level hierarchy for Gas Trades Breakdown output elements
GAS_TRD_BRKDN_QRY	Top level hierarchy for Gas Trades Breakdown query elements
GAS_TRD_DTL	Top-level hierarchy for Gas Trade details.
HDR_DTL	Top-level Header details.
HR_BR	Hour bar
HSTRY_DTL	Top level hierarchy for the History Detail Elements
I_O_IND	Input Output Indicator (a flag to indicate flow direction)
IGNR_TLRNCE	Ignore Tolerance
INS_NM_DTL	Top-level hierarchy for INS Nomination Headers.
INS_NOM	The value of the INS Nomination
INTR_METER_NO	The interruptible meter name for the storage output meter.
IP_BAL	The total amount of energy requested to be flowed into the NBP
LCN	Meter ID.
LCTN	Location
LCTN_CD	Location (Zone) Description
LCTN_DESC	Location (Zone) Identifier
LCTNS	Locations
LDZ	The LDZ Id

LDZ_NM	The LDZ name
LNPK_CMNT	Additional details related to the Status Information are entered here
MATCH_YN	Matching trades flag Matched- Yes Unmatched- No
MAX_ACPTD_PRC	Maximum Accepted Price
METADATA	Top level hierarchy for meta information elements
METER_ID	Meter Id for which storage output claims are retrieved
METER_TYPE	Specifies Type of meter
MIN_ACPTD_PRC	Minimum Accepted Price
MIN_QTY	Minimum Quantity
MKT_OP_CD	Market Operator for whom Trade Breakdown details are retrieved.
MOS	Method Of Sale
MSG	The Message to be shown to BA
MSG_CD	Message Code
MSG_DESC	Message Description
MSRD_QTY	Quantity of energy measured for the specified Meter ID/Gas day.
MTR_ID	Meter Id
MTR_NAME	Meter Name
MTR_TYPE	Meter Type
MTR_ZON_DTL_HDR	Meter Zone Relationship Header Details
MTR_ZON_DTLS_INF	Meter Zone Relationship Information Details
NA_RNM_UPDTD	This shows whether the Renomination is created for the NDMA meters of the shipper. If the percentage change of the Total NDMA Forecast is greater than or equal to the set percentage value, then this indicator will be set to 'Y'.
NDM	The energy details for Non-Daily Meters
NDMA_OP_RNM_FCAST	NDMA output renomination forecast for the BA, for the gas day
NET_BAL	The difference between the requested input balance and requested output balance
NET_CURR_ALLOC_QTY	Indicates the net current allocated quantity
NET_NMTD_QTY	Indicates the net nominated quantity
NET_REQ_NRG	The difference between the requested input and the requested output energy
NET_SCHD_NRG	The difference between the scheduled energy bought and sold.
NM_SYS_HSTRY_OUT	Top level hierarchy for output elements
NM_SYS_HSTRY_QRY	Top level hierarchy for query elements
NO_ACPTD_BIDS	No. of Accepted Bids
NO_BID_SHPRS	No. of Bidding Shippers
NO_OF_ACTVTS	Number of activities
NO_OF_STPS	Number of renomination steps
NO_SUCS_SHPRS	No. of Successful Shippers
NRG_DTL	Top-level hierarchy for energy Details.
NRG_FCAST_DTL	Top-level hierarchy for Shipper Total Energy Forecast Headers.
OBO	On behalf of
OCM_NBP_BUYS	Sum of all Title Swap buy trades for a BA
OCM_NBP_SELLS	Sum of all Title Swap sell trades for a BA
OCM_PHY	Sum of all OCM Physical Nominations (Buy or Sell)

	for a BA
OP_BAL	The total amount of energy requested to be flowed out of the NBP
OPN_LNPK	The linepack at the start of the gas day (Mcm)
OPP_FL	Opposite flow indicator
OPP_FL	Opposite flow indicator
PCLP1	The Projected Closing Linepack 1 (Mcm)
PCLP2	The Projected Closing Line pack 2 (Mcm)
PCTG	Tolerance percentage set for Each Meter Type.
PR_INFO_HSTRY_DTL	Top level hierarchy for the Individual record elements
PR_INFO_HSTRY_OUT	Top level hierarchy for output elements
PR_INFO_HSTRY_QRY	Top level hierarchy for query elements
PRC	The price at which the energy was sold/purchased
PRCNT_CHNG	Percentage change
PRD_FRM	Period From
PRD_TO	Period To
PRDT	Product
PRM_PRC	Premium Price
PROJ_EOD_IMBAL	The projected day imbalance for INS Nomination
PRTY_CD	Party Code of the BA
QTY_BGT	Quantity Bought Back
QTY_NOT_BGT	Quantity Not Bought Back
QTY_SLD	Sold Quantity
QTY_UNSLD	Unsold Quantity
REG_DTL	Top level hierarchy for set of registered records.
REG_PHY_LOC_DTL	Top level hierarchy for individual elements
REG_PHY_LOC_OUT	Top level hierarchy for output elements
REG_PHY_LOC_TRD	Top level hierarchy for individual input records
REG_PHYS_LOC_TRDS	Top level hierarchy for Register Physical/Locational Trade elements
REG_PHYS_LOC_TRDS_DTL	Top level hierarchy for individual input records
REG_TTL_SWP	Top level hierarchy for individual input records
REG_TTL_SWP_DTL	Top level hierarchy for individual elements
REG_TTL_SWP_OUT	Top level hierarchy for output elements
REG_TTL_SWP_RGSTR	Top level hierarchy for Register Title Swap elements
RENOM_DTL_INFO	The detail record for renomination
REQ_INP_NRG	The requested input energy value at the specified hour bar.
REQ_NRG	The amount of energy nominated by the shipper
REQ_NRG_DTL	Top-level hierarchy for requested energy details.
REQ_NRG_TOT	The sum of the requested energies
REQ_OP_NRG	The requested output energy value at the specified hour bar
REQ_STS	Requested status
RNM_DTL	Top-level hierarchy for renominations details
RNM_DTL_INF	The detail record for Renomination
RNM_DTLS_INF	The detail record for the renomination
RNM_HDR	Top-level hierarchy for renominations headers.
RNM_INF	The detail record for renomination info
RSN	Reason for modification of a record
RSN_CD	Reason for rejecting the Bid.
RSN_FOR_RJCTN	Reason For Rejection/pro-rating the capacity
RT_SCHD	Type of BA service
RVNU	Revenue
SAP	System Average Price (p/kWh)

SCHD_INP_NRG	The scheduled input energy value at the specified hour bar.
SCHD_NRG	Energy approved by the O&T against the nominated value.
SCHD_NRG_BGT	The approved energy bought.
SCHD_NRG_DTL	Top-level hierarchy for scheduled energy details.
SCHD_NRG_SELL_STS	The Status of the approved energy sold.
SCHD_NRG_SMRY	Top-level hierarchy for scheduled energy Summary.
SCHD_NRG_SOLD	The approved energy sold.
SCHD_NRG_SUMM	Top-level hierarchy for Scheduled Energy Summary.
SCHD_NRG_TOT	The sum of the scheduled energies
SCHD_OP_NRG	The scheduled output energy value at the specified hour bar
SCHD_STS	Scheduled status
SCLNG_FCTR	Scaling factor
SELL_DTL	Top-level hierarchy for Sell details.
SERVC_ID	Unique identifier for a Shipper's service
SF	Special function
SHCD_NRG_BUY_STS	The status of the approved energy bought.
SHPR	The BA's Abbreviated Name.
SHPR_EOD_NMS_OUT	Top level hierarchy for output elements
SHPR_EOD_NMS_QRY	Top level hierarchy for query elements
SHPR_PRLIM_BAL_OUT	Top level hierarchy for query elements
SHPR_PRLIM_BAL_QRY	Top level hierarchy for meta information elements
SHPR_TOT_NRG_FCAST_OUT	Top level hierarchy for output elements
SHPR_TOT_NRG_FCAST_QRY	Top level hierarchy for query elements
SHPR_TRD_DTL	Top-level hierarchy for Shipper Trade details.
SHPR_TRD_DTL_OUT	Top level hierarchy for Shipper Trade details Output elements
SHPR_TRD_DTL_QRY	Top level hierarchy for Shipper Trade details query elements
SMP_BUY	System Marginal Price Buy (p/kWh)
SMP_SELL	System Marginal Price Sell (p/kWh)
STEP	Top level hierarchy for step elements
STEP_NRG	Step energy
STR_TIME	Start time
STRG_IP	Sum of all input nominations at storage meters for a BA
STRG_OP	Sum of all output nominations at storage meters for a BA
STRG_OP_CLMS_OUT	Top level hierarchy for Storage Output Claims output elements
STRG_OP_CLMS_QRY	Top level hierarchy for Storage Output Claims query elements
STRG_OP_DTL	Top-level hierarchy for Storage output details.
STRT_ACTVTY_NBR	Start activity number
STRT_TM	Start time
STS	Status (Accepted/Rejected) of the Trade
STS_OUT	Top level hierarchy for output elements
SUBTNDR_ID	Sub Tender ID
SYS_BAL_NOM_OUT	Top level hierarchy for System Nomination Balance Output elements
SYS_BAL_NOM_QRY	Top level hierarchy for System Nomination Balance query elements
SYS_STATS_HSTRY_DTL	Top level hierarchy for the Individual record elements

SYS_STS_INFO	Top level hierarchy for the record elements
TLRNCE_FLG	Flag to indicate as to whether the tolerance check needs to be carried out or ignored
TLRNCE_QTY	Tolerance quantity for the meter type
TMSTMP	Timestamp
TMSTMP_MAX_PRC	Timestamp of allocation of bid with the maximum accepted price
TMSTMP_MIN_PRC	Timestamp of allocation of bid with the minimum accepted price
TOT	Total
TOT_ALLOC_CPCTY	Total Allocated Capacity
TOT_ALLOC_QTY	Indicates the total net current allocated quantity.
TOT_BID_CPCTY	Total Demanded Bid Capacity
TOT_CLM_QTY	Indicates the total claimed quantity.
TOT_FCAST	Shipper's Total NDMA forecast across all the LDZs for the selected gas day
TOT_IP	Total input quantity
TOT_MIN_QTY	Total Minimum Quantity
TOT_MSMT_ALLOC	Total allocated/measured quantity for the meter type
TOT_OP	Total output quantity
TOT_SCHD_NRG_BGT	The sum of all the approved energy bought by all Shippers.
TOT_SCHD_NRG_NET	The sum of net energy of each Shipper.
TOT_SCHD_NRG_SOLD	The sum of all the approved energy sold by all Shippers.
TRD_BRKDN_DTL	Top-level hierarchy for Gas Trade Breakdown details.
TRD_BUY	Sum of all OTC NBP buy trades for a BA
TRD_PTNR	The Shipper Abbreviated Name involved in the Trade.
TRD_QTY	Total energy bought/sold by the BA
TRD_SELL	Sum of all OTC NBP sell trades for a BA
TRD_SUMM_DTL	Top-level hierarchy for Trade Summary details.
TRD_TYP	Trade Type
TRLR_DTL	Top-level hierarchy for Trailer details
TRNS_IP	Sum of all nominations at input meters (other than storage input meters) for a BA
TX_END_DT	Transaction End Date
TX_STRT_DT	Transaction Start Date
UPDT_DT	Updated date
UPDT_INS_NMS	Top level hierarchy for update elements
UPDT_INS_NMS_DTLS	Top level hierarchy for update detail elements
UPDT_INS_NMS_OUT	Top level hierarchy for update elements
UPDT_RNM	Top level hierarchy for the Update Renominations API
UPDT_RNM_DTL	Top level hierarchy for the details of update Renominations
UPDT_RNM_OUT	Top level hierarchy for update elements
UPDT_TM	The latest time when the System Status details were updated.(hh:mm:ss)
VOL	Volume
VOL_BGT_MAX_PRC	Volume Bought at Maximum Price
VOL_BGT_MIN_PRC	Volume Bought at Minimum Price
VOL_REQ_MAX_PRC	Volume Requested at Maximum Price
VOL_REQ_MIN_PRC	Volume Requested at Minimum Price
VOL_SLD_MAX_PRC	Volume Sold at Maximum Price
VOL_SLD_MIN_PRC	Volume Sold at Minimum Price
VW_INS_NMS_OUT	Top level hierarchy for INS Nomination elements

VW_INS_NMS_QRY	Top level hierarchy for INS Nomination query elements
VW_RNM_DTLS_OUT	Top level hierarchy for output elements
VW_RNM_DTLS_QRY	Top level hierarchy for query elements
VW_RNM_OUT	Top level hierarchy for output elements
VW_RNM_QRY	Top level hierarchy for query elements
VW_WCF_SF_VAL_DTL	Top level hierarchy for the Individual record elements
VW_WCF_SF_VAL_OUT	Top level hierarchy for output elements
VW_WCF_SF_VAL_QRY	Top level hierarchy for View WCF/SCF Values elements
WAP	Weighted Average Price
WAP_TOP50_ALOC_QTY	WAP of Top 50% of Allocated Quantity
WCF	Weather correction factor

7 Hints and Tips

This section contains various miscellaneous hints and tips for API client developers.

- If you are debugging a login API client then you may make several unsuccessful attempts to login. It is useful to intersperse these attempts with a login to the Gemini screen service via Citirx. This achieves two things:
  1. It confirms that your login credentials are correct and any failure to login via the login API is down to something else.
  2. It resets your consecutive unsuccessful login attempts count for that userid. This is important to prevent the account being locked.

  Be aware that when you login to the Gemini screens via an API userid you may see an error to the effect that the application has failed to build a menu. This can be ignored for userids with API access only. It indicates that no screens are allocated to the userid.
- During API client development use the XSDs on the Gemini servers to validate your request XML before sending it. There are various third party tools that will validate XML against an XSD.
- Avoid physically storing the request/response XML unless you have to. Input/output operations are resource expensive.
- DOM parsers are more memory intensive for read operations. If you have the choice, use SAX parsers for read and DOM for generate/update. SAX is much quicker than DOM for read operations.
- The error XML contains the error code and associated error messages. If you would rather display a different message you can map our error codes to your own specific messages.
- Remember that API sessions will time expire after sixty minutes of inactivity. If you client is inactive for this period you will have to login again to re-establish a session.
- Make the URLs that you access configurable. Do **not** hard code them within your API clients.

  This is especially important for the root URL of the API service. Changes to domain naming policies, for example, can affect these URLs. For this reason we specify URLs in all documentation relative to the root URL of the API service, designated by a single "/".

  It is useful to separately hold the root URL of the API service that you are accessing, which will be common to all API calls, and the relative URLs, which will be specific to each individual API.
- API clients must not be multi-threaded.
- To diagnose errors try to trap your raw HTTP responses and requests. APIs are intended to be client language independent and the requirements for API client

interaction with the API are deliberately specified terms of HTTP and associated XML structures.

If you are investigating a suspected API error then it will greatly assist both yourselves and us to see the interaction in terms of HTTP messages. What happens in various client languages to send/receive those messages might be very different.

- If you schedule changes of password via the change password API before your password expires, then you avoid the complexity of having to react to a password expiry warning or actual expiration. Current passwords will expire after thirty days of use. You are warned five days in advance.

8 Document Control

8.1 *Superseded Documents*

Title	Reference	Version	Date
Gemini – API Specification for Business Associates (Entry Capacity Releases)	APISpecificationEntryCapRel	2	18-Mar-2004

8.2 *Version History*

Version	Status	Date	Author(s)	Summary of Changes
1	Approved	20-Jan-2005	NGT IS	Initial version. Prior history removed for brevity.
2	For Representation	28-Jan-2005	NGT IS	- Major revision for greater clarify following feedback from API developer workshops. - Addition of change password API. - Addition of facility to compress response XML from APIs. - Addition of references to additional functional APIs added to Gemini scope.
2	For Representation	15-Feb-2005	NGT IS	Changes following NGT internal review.
2	For Approval	11-Mar-2005	NGT IS	- Heading 5.1.2 added - Correction to 3.4.5 Sample Code
2	Approved	17-Mar-2005	NGT IS	Approved at UK-Link Committee.
3	For Representation	07-Sep-2005	National Grid IS	Corrections highlighted during Gemini trials as follows: - Missing “s” inserted into change password API URL in section 3.4; - Missing home/ added to the path of the logout API URL in section 3.3; - Various corrections to the table in section 5.1.2; - Redirection guide added as section 3.1.3. Also, references to NGT changed to National Grid.
3	For Approval	29-Sep-2005	National Grid IS	No representation comments received. Submitted “For Approval” without further change.