HIGHER EDUCATION INFORMATION MANAGEMENT SYSTEM (HEIMS)

HEIMS WEB SERVICES INTERFACE

TECHNICAL SPECIFICATION APPLICATIONS AND OFFERS FUNCTIONS
Table of Contents

1  Introduction ................................................................................................................................. 3
  1.1  Purpose of this document ........................................................................................................... 3
  1.2  Out of Scope ............................................................................................................................... 3
  1.3  Target Audience .......................................................................................................................... 3
  1.4  Disclaimer .................................................................................................................................. 3
  1.5  Version Control ........................................................................................................................... 3

2  HEIMS Technical Architecture ........................................................................................................ 4
  2.1  Web Services Architecture Overview .......................................................................................... 4
  2.2  Technical Requirements ............................................................................................................. 4
  2.3  Compression ............................................................................................................................... 5
  2.4  Security ..................................................................................................................................... 7
  2.5  Transactions Overview ............................................................................................................... 7
    2.5.1  Batch Transactions ................................................................................................................. 7
  2.6  Service Agreements and Availability ......................................................................................... 8

3  XML Schemas ................................................................................................................................ 9
  2.7.1  LoadApplicationsAndOffersSubmissionIn ............................................................................. 10
  2.7.2  LoadSubmissionOut ................................................................................................................ 11
  2.7.3  RequestControlTable ............................................................................................................. 11
  2.7.4  ResponseControlTable .......................................................................................................... 12

4  Request and Response Definitions .................................................................................................. 12
  2.8.1  Request Identifiers ................................................................................................................ 12
  2.8.2  Request Object Schema ......................................................................................................... 13
  2.8.3  Response Schema .................................................................................................................. 13
  2.8.4  Control Data .......................................................................................................................... 14
  2.8.5  Transaction Data .................................................................................................................... 15
  2.8.6  Messages ................................................................................................................................ 16

5  Error Handling ............................................................................................................................... 17
  2.9.1  Schema Errors ....................................................................................................................... 17
  2.9.2  Business Errors ...................................................................................................................... 18

3  Interfaces ...................................................................................................................................... 19
  3.1  General Information .................................................................................................................. 19
  3.2  LoadApplicationsAndOffersSubmission method (for submission of Applications and Offers data for batch processing) ........................................................................................................ 19
  3.3  GetApplicationsAndOffersSubmissionResults method (for retrieving the results of a processed submission of Applications and Offers data) ........................................................................... 20
  3.4  Ping method ............................................................................................................................... 20

4  Security – Change Password .......................................................................................................... 21

5  Environments ................................................................................................................................. 22
  5.1  Production .................................................................................................................................. 22
  5.2  Next Production Release ............................................................................................................ 22

6  Certificate ..................................................................................................................................... 22

7  Troubleshooting ............................................................................................................................. 22

A. Appendix A – References and Other Relevant Documents ............................................................ 23
  A.1  References .................................................................................................................................. 23
  A.2  Other Relevant Documents ........................................................................................................ 23

B. Appendix B – Glossary .................................................................................................................. 24

C. Appendix C - Data Element Map ................................................................................................... 26

D. Applications and Offers Web Service Requests and Responses .................................................... 31
  D.1  LoadApplicationsAndOffersSubmission .................................................................................. 31
  D.2  GetApplicationsAndOffersSubmissionResultsRequest ............................................................ 32
1 Introduction

1.1 Purpose of this document
The purpose of this document is to provide technical information required to make Web Service calls from Tertiary Admissions Centres (TACs) systems to the department’s Higher Education Information Management System (HEIMS). The scope of the document includes technical specifications of the Web Services for Applications and Offers submissions, descriptions of the available web methods, details of the schemas used and troubleshooting information.

Due to the flexibility of Web Services and the large number of systems that can consume them, it is not feasible to provide detailed implementation instructions for specific platforms and systems. This document will however, provide what technical information is needed in order to configure and use a Web Services toolkit.

1.2 Out of Scope
The following are out of scope for this document:
- Detailed business rules used to validate Tertiary Admissions Centre submission data; and
- Technical specification for any of the W3C standards used in HEIMS.

A list of documentation that addresses W3C standards and context information for HEIMS Web Services (such as business requirements) can be found in Appendix A.

1.3 Target Audience
This is a technical document for use by Tertiary Admissions Centres (TAC Developers).

1.4 Disclaimer
These specifications give information about how to use the HEIMS Applications and Offers Web Services. These specifications are not intended to provide implementation instructions for individual systems. The department accepts no responsibility for any loss or damage to any system resulting from the use of these specifications.

These specifications may be changed from time to time. It is the responsibility of TACs and others using these specifications to ensure they are using the latest version.

Technical enquiries relating to these specifications can be initially directed to the HEIMS Data Collections at HEIMS.datacollections@education.gov.au or phone on (02) 6240 7487.

1.5 Version Control
Documenting changes to this document will be managed through controlled versioning. Updated versions of this document can be obtained by contacting the HEIMS Data Collections at HEIMS.datacollections@education.gov.au or phone on (02) 6240 7487.
2 HEIMS Technical Architecture

2.1 Web Services Architecture Overview
XML Web Services provide a mechanism for applications to exchange information over a network. By providing a standard interface and communicating using international standard protocols, all Web Service implementations operate in the same manner making communicating using Web Services a simple, open and platform independent process.

Web Service interfaces are described by the Web Services Definition Language (WSDL) and Web Service communication relies on protocols such as Transmission Control Protocol/Internet Protocol (TCP/IP), Hypertext Transfer Protocol (HTTP), and Simple Object Access Protocol (SOAP). These standards are developed and maintained by the World Wide Web Consortium (W3C), a member organisation consisting of leading technology vendors, corporate users, standards bodies and government organisations. As such, this provides a set of rich standards which are not tied to a specific vendor or system. For these reasons, XML Web Services are suitable as a means for HEIMS to communicate with external systems.

2.2 Technical Requirements
HEIMS Web Services will only cater for requests using SOAP 1.1. Web Service calls using HTTP-GET and HTTP-POST will not be supported. This is because using SOAP provides XML schema support for more complex data types. The transport method supported is Transport Layer Security (TLS1.2 and above).

In order to call the HEIMS Web Services, an application or SDK capable of calling XML Web Services is required. The application or SDK must support the following:

- XML 1.0;
- SOAP 1.1;
- HTTP 1.1;
- Basic Authentication;
- Server Name Identification.

When implementing a responsive application, bandwidth requirements must be taken into consideration. Bandwidth requirements depend on many factors. These include:

- Size of the payload sent to HEIMS for processing;
- Frequency of the requests; and
- Data compression.

HEIMS Applications and Offers Web Services have been designed to minimise the network traffic payload as much as possible. The services provided by HEIMS are therefore not bandwidth intensive. However, to ensure best performance, the department recommends a broadband connection for both upstream and downstream traffic.
2.3 Compression

Compression is a standard feature in HEIMS Web Services. All HEIMS Applications and Offers Web Service requests and responses will be compressed. For clients with small bandwidth capabilities or those that process large volumes of requests, this will be a useful feature as the bandwidth savings can be quite large. Using compression reduces not only bandwidth requirements, but other side effects of large requests such as timeouts. The compression algorithm supported by all HEIMS Web Services is gzip.

In order to use compression, the tools used to call HEIMS Applications and Offers Web Services must be able to control how and when SOAP messages are created and sent. Compression of the stream must take place in the SOAP layer. That is, after the SOAP envelope has been properly constructed, the body of the SOAP envelope must be compressed and then sent over HTTP to the server.

To inform the server that the request is compressed:

- The following Compression header must be added to the SOAP header

```xml
<Compression
  s:mustUnderstand="1"
```

<table>
<thead>
<tr>
<th>Compression Header Attribute</th>
<th>Value Required</th>
</tr>
</thead>
</table>

- And the compressed content of the Soap Message Body must be enclosed within these elements:

```xml
  ...
</SC:CompressedData>
```

Compression Example (Please note that abbreviated dummy data has been used and the schema may not be accurate):

Before Compression:

```xml
<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
  <s:Body>
    <LoadApplicationsAndOffersSubmission
      xmlns="http://dest.gov.au/Heims/Service">
      <request
        xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:RequestControlData>
          <a:RequestId>128695597818308750</a:RequestId>
          <a:ClientOrganisationCode>5003</a:ClientOrganisationCode>
          <a:RequestLocalDateTime>2008-10-27T16:43:01.830875+11:00</a:RequestLocalDateTime>
        </a:RequestControlData>
        <a:TransactionData>
          <a:LoadApplicationsAndOffersSubmissionIn>
            <a:RecordId>0</a:RecordId>
            <a:Application>
              <a:ApplicationIdentificationCode>APP0</a:ApplicationIdentificationCode>
              <a:ApplicationStatusCode>1</a:ApplicationStatusCode>
              <a:ApplicantName>
                <a:FamilyName>Aaaaaaa</a:FamilyName>
                ...
              </a:ApplicantName>
              <a:Offers>
                <a:Offer>
                  <a:SectorCode>1</a:SectorCode>
                  <a:PreferenceOrdinalPosition>00</a:PreferenceOrdinalPosition>
                </a:Offer>
              </a:Offers>
            </a:Application>
          </a:LoadApplicationsAndOffersSubmissionIn>
        </a:TransactionData>
      </request>
    </LoadApplicationsAndOffersSubmission>
  </s:Body>
</s:Envelope>
```
After Compression:

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
      H4sIAAAAAAAAEO29BSvVK1+B0o... ...G24Z2PtE1
    </SC:CompressedData>
  </s:Body>
</s:Envelope>
```

If the server detects the SOAP Message header with Compression element with correct associated attributes (as specified above), it will treat the request stream as being compressed. The server will then uncompress the body stream enclosed within the <SC:CompressedData> </SC:CompressedData> tags into the original SOAP message body and process it as normal. If the server can not find the SOAP Message HTTP header with Compression element with correct associated attributes, an exception will be thrown to the client.

To inform the Client that the response is compressed:

- The following Compression header will be added to the SOAP header

```xml
<Compression
  s:mustUnderstand="1"
```

<table>
<thead>
<tr>
<th>Compression Header Attribute</th>
<th>Value Required</th>
</tr>
</thead>
</table>

- And the compressed content of the Soap Message Body will be enclosed within these elements:

```xml
  <SC:CompressedData>
```

The client will have to intercept the response from the Server and will have to uncompress the stream with in the <SC:CompressedData> </SC:CompressedData> into the original SOAP message body and process it as normal.
2.4 Security
HEIMS Applications and Offers Web Services use Basic Authentication over SSL for secure communication. Basic Authentication is widely supported and part of the HTTP 1.1 standard. In Basic Authentication, the client provides the username and password, which are Base64 encoded and sent directly to the server. Encryption of the communication channel is then handled by the SSL protocol.

The following security rules apply to login passwords for HEIMS Web Services:

- Passwords will expire after 30 days;
- The last 10 passwords are recorded so that passwords cannot be reused;
- Strong passwords will be used – They must contain a combination of upper and lowercase characters, numbers and special characters (eg. #, @, $);
- Password must be between 7 and 15 characters long;
- After a password has been successfully changed by a user, it cannot be changed again within a 24-hour period except by the HEIMS IT Liaison officer.
- The maximum number of failed logon attempts before the account will be locked is 3;
- If the account is locked, the HEIMS Help Desk (HEIMS.datacollections@education.gov.au or phone on (02) 6240 7487 must be contacted to reset the password.
- After a password has been reset, it must be changed by calling the ChangePassword method before any other HEIMS Web methods are called. The number of days elapsed since the password was changed is then reset to zero.

2.5 Transactions Overview
The LoadApplicationsAndOffersSubmission method of the Applications and Offers Web Service has only a batch implementation. Any LoadApplicationsAndOffersSubmission request will be placed in the batch queue on the server to be processed when resources become available on the server.

2.5.1 Batch Transactions
Batch transactions are submissions that are placed in a queue for later processing. Batch methods can contain requests for multiple transactions and therefore can be very large in size.

An invocation of a batch transaction will usually comprise of two method calls: The first call contains any request-specific information plus any transaction data. The server accepts the request and marks it for processing. The second (and optional) call polls the server and checks for the results of the job. If the job has completed, the results are returned back to the caller. Otherwise, the server will inform the client that the original request is still being processed.

![Figure 1: Message Flow in Batch Transactions](image-url)
2.6 Service Agreements and Availability

The department is committed to providing a high performance, reliable, available and fully supported HEIMS production environment. The department will endeavour to provide the following response times to all approved HEIMS users:

- **Batch Transactions** – 95% of all batch submissions will be processed within 24 hours after receipt by the department.

These response figures do not include network transport times. In other words, the response times given are from the time the department servers receive the request to the time the results are sent back from the the department servers.

The daily limits start at 12 midnight HEIMS server time (Canberra) and apply for the following 24 hours, rather than conforming to a 24 hour rolling window.

It is intended that HEIMS Applications and Offers Web Services will be available 24 hours a day, 7 days a week with the following possible exceptions:

- A weekly infrastructure maintenance window on Thursday evenings, 7.00pm-12.00am AEST/AEDT. HEIMS Applications and Offers Web Services will generally be available during this period but interruptions to service availability may occur;
- A HEIMS system maintenance window as required but on Saturdays with 1 week notification by the department. System availability in this period will depend on the amount of production maintenance required.

The department will endeavour to minimise all system maintenance during peak enrolment periods and keep unscheduled down-times to an absolute minimum.
2.7 XML Schemas

This section describes the top level XML Schema complex types used in Applications and Offers Web Service Request and Responses.

The actual request and response types are described in section 2.8. This section will be used to show the high-level business objects that can be modified directly through Applications and Offers Web Services. Full schema definitions for all HEIMS data types can be found in the WSDL (See section 3.1). Types referenced in this section that are not represented by complex types are actually simple types.

XSD elements are used everywhere instead of attributes for compatibility reasons with older SDKs that will be used to bind data serialisation code to the schemas.
2.7.1 LoadApplicationsAndOffersSubmissionIn

LoadApplicationsAndOffersSubmissionIn refers to transaction data that contains information about applications for admission to a higher education undergraduate award course submitted through a Tertiary Admissions Centre (TAC). It includes details about the applicants, preferences and offers made.

LoadApplicationsAndOffersSubmissionIn is part of the LoadApplicationsAndOffersSubmissionInRequest.
2.7.2 LoadSubmissionOut

LoadSubmissionOut contains the input transaction data as a stream and the status of the transaction data. LoadSubmissionOut is part of the LoadApplicationsAndOffersSubmissionResponse.

Figure 3: LoadSubmissionOut XSD complex type

Note: Input Stream is a base64binary stream. Input Stream contains input transaction data of type LoadApplicationsAndOffersSubmissionIn which is used for providing Applications and Offers details. Refer to section 2.7.1 for more information on LoadApplicationsAndOffersSubmissionIn.

2.7.3 RequestControlTable

RequestControlTable is a container for all request control data. For every call to an Applications and Offers Web Service, a RequestControlTable must always be sent. The RequestControlTable is sent wrapped within a Request object.

Figure 4: RequestControlTable XSD complex type
2.7.4 ResponseControlTable

ResponseControlTable is a container for all response control data. When a client receives a response from an Applications and Offers Web Service, a ResponseControlTable is always present. The ResponseControlTable is sent wrapped in a response object.

---

![ResponseControlTable XSD complex type](image)

Figure 5: ResponseControlTable XSD complex type

2.8 Request and Response Definitions

The following object types are used generally in all HEIMS Web Service calls:

- Response objects and request objects;
- Response control tables and Request control tables; and
- Response transaction data and request transaction data.

Note: In this and following sections, element fields will be defined in terms of XML data types. If an XML data type has a prefix of `xs`, it refers to a built-in XML data type that is part of the [http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema) namespace. If the data type has a `heims` prefix, it refers to elements that are part of the [http://dest.gov.au/Heims/](http://dest.gov.au/Heims/) namespace. Full schema definitions for all HEIMS data types can be found in the WSDL (see section 3.1).

2.8.1 Request Identifiers

Request identifiers (RequestIds) are an important part of HEIMS Applications and Offers Web Services. They are included in every web method call to identify an individual request. The length of the RequestId must be between 1 and 36 characters. The suggested format of the RequestId is "HHHAO YYYYMMDD HH:MM:SS:NN" where:

- "HHHH" is your TAC Client Organisation Code
- "AO" is the literal text "AO"
- "YYYY" is the four digit current year
- "MM" is the two digit current month
- "DD" is the two digit current day
- "HH" is the two digit current hour
- "MM" is the two digit current minute
- "SS" is the two digit current second
- "NN" is the two digit current millisecond

The value of the RequestId must only be unique within a particular TAC and it must retain uniqueness over time. It is the responsibility of the TAC to ensure that all its client systems internally keep track of all previous RequestIds and to synchronise generation of unique RequestIds for new calls.

Every time a method is called, a new RequestId must be sent to the server. When returning the processing results of the request, the server response will include the same RequestId. In order to guarantee that the server does not process the same request twice due to a communication failure, calls to a web method providing an old RequestId will return the results of the old request. No new processing will be performed.
2.8.2 Request Object Schema

The request object is a generic container that holds all the information a client needs to send to a HEIMS Applications and Offers Web Service. It contains request control data (one `RequestControlTable`) plus multiple request transaction data elements. For batch methods, there is no limit on the number of transactions allowed per request.

The classes shown in Figure 6 below represent all of the request classes.

![General Request Class Diagram](image)

Figure 6: General Request Class Diagram

The classes in the diagram above represent complex types in the XML Schema. Other types have a simple mapping to XSD simple types, such as `DateTime (xs:dateTime)` and `string (xs:string)`. Enumerations map trivially to XSD enumerations. Note that the List<T> type (used in each request class) will be mapped to `ArrayOf{T}` as a SOAP operation parameter.

2.8.3 Response Schema

Similarly, the response schema is a container for all information an Applications and Offers Web Service sends to a client. It contains response control data (one `ResponseControlTable`) plus one or more response transaction data elements.

The classes shown in the diagram below represent all of the response classes.

![General Response Class Diagram](image)

Figure 7: General Response Class Diagram
The classes in Figure 7 above represent complex types in the XML Schema. Other types have a simple mapping to XSD simple types, such as `DateTime (xs:dateTime)` and `string (xs:string)`. Enumerations map trivially to XSD enumerations. Note that the `List<T>` type (used in each response class) will be mapped to `ArrayOf{T}` as a SOAP operation parameter.

2.8.4 Control Data

Control data is present in all communication between clients and HEIMS Applications and Offers Web Services. It provides information concerned with the HEIMS Applications and Offers Web Service infrastructure and as such, is essential to every HEIMS Applications and Offers Web method. There are two control data structures: `RequestControlTable` and `ResponseControlTable`.

RequestControlTable

`RequestControlTable` is a container for all request control data. For every call to a HEIMS Applications and Offers Web Service, a `RequestControlTable` must always be sent. It must be sent wrapped within a request object. The `RequestControlTable` contains the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestId</td>
<td>heims:RequestId</td>
<td>The request identifier.</td>
</tr>
<tr>
<td>ClientOrganisationCode</td>
<td>heims:ClientOrganisationCode</td>
<td>The client’s 4-digit identifying code.</td>
</tr>
<tr>
<td>RequestLocalDateTime</td>
<td>xs:dateTime</td>
<td>Date and time on the client at the time of the call.</td>
</tr>
</tbody>
</table>

ResponseControlTable

Similarly, the `ResponseControlTable` is a container for all response control data. When a client receives a response from a HEIMS Applications and Offers Web Service, a `ResponseControlTable` is always present. It must be sent wrapped in a response object. A `ResponseControlTable` object contains the following fields:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestId</td>
<td>heims:RequestId</td>
<td>The same request identifier of the original request.</td>
</tr>
<tr>
<td>ClientOrganisationCode</td>
<td>heims:ClientOrganisationCode</td>
<td>The 4-digit identifying code of the client that issued the original request.</td>
</tr>
<tr>
<td>CurrentDateTime</td>
<td>xs:dateTime</td>
<td>Date and time on the server at the time the response was sent.</td>
</tr>
<tr>
<td>ReceivedDateTime</td>
<td>xs:dateTime</td>
<td>Date and time on the server when the server first received the request.</td>
</tr>
<tr>
<td>RequestStatus</td>
<td>heims:RequestStatus</td>
<td>Contains a <code>RequestStatusCode</code> – the current status of the request, and <code>Messages</code> – a list of messages associated with the request.</td>
</tr>
</tbody>
</table>

The `RequestStatus` element contains two fields, `RequestStatusCode` and `Messages`. `RequestStatusCode` contains information on the status of the request itself. It is not related to the status of the individual transactions. For example, a request can have a `RequestStatusCode` equal to ‘SUCCESS’, but all of its individual transactions can fail business processing. `RequestStatusCode` can take the following values:

<table>
<thead>
<tr>
<th>RequestStatusCode</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUPLICATE</td>
<td>The <code>RequestId</code> of the current request has already been used. Request transaction data will contain the results of the original request. No new processing was performed on the server.</td>
</tr>
<tr>
<td>FAILURE</td>
<td>An error occurred with the request. Request transaction data will not contain results.</td>
</tr>
<tr>
<td>PROCESS</td>
<td>The request is currently being processed or is queued for processing. Request transaction data will not contain results.</td>
</tr>
<tr>
<td>ARCHIVE</td>
<td>The <code>RequestId</code> of the current request has already been used more than 30 days ago. The server has since then archived the processing results. Request transaction data will not contain results.</td>
</tr>
<tr>
<td>SUCCESS</td>
<td>The request has been processed successfully. Request transaction data will contain processing results.</td>
</tr>
<tr>
<td>SUBMITTED</td>
<td>The request has been successfully submitted for batch processing.</td>
</tr>
</tbody>
</table>
The Messages element contains a list of messages concerned with the request. For example, if RequestStatusCode has a value of ‘FAILURE’, Messages will contain descriptions as to why the original request was invalid.

### 2.8.5 Transaction Data

Transaction Data contains business information specific to the method being called. Request Transaction Data contains data required to perform the required business service, while SubmissionResults (Response Transaction Data) contains the results of processing the request.

Request TransactionData is currently being used in LoadApplicationsAndOffersSubmissionRequest only. SubmissionResults (Response Transaction Data) is only used in DataSubmissionResponse.

Because each method has different input and output fields, each method defines different request and response transaction data schemas. There are however, two fields that are common across transaction schemas, RecordId and TransactionStatus.

**Record Identifier**

The record identifier (RecordId) is an element common to both request and response transaction data elements. When a batch request contains more than one transaction, the RecordId is used to distinguish between each transaction data element.

The RecordId is an integer. For any request, the value of the RecordId must be unique across all transactions in the request.

In the context of a response transaction data element, the RecordId will match that of the corresponding request transaction data element. This allows the client to identify which output record matches up with which input record.

**TransactionStatus**

TransactionStatus is an element common only to response transaction schemas. It provides information about the results of business processing for an individual transaction. TransactionStatus contains three fields: TransactionStatusCode, RequestId and Messages. TransactionStatusCode indicates the processing status of the transaction, RequestId indicates which request this transaction belongs to and Messages is an array of Message elements. Within the context of TransactionStatus, each Message element holds business validation or business error messages for that transaction.

TransactionStatusCode can take on the following values:

<table>
<thead>
<tr>
<th>TransactionStatusCode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCCESS</td>
<td>The individual transaction processed correctly according to its respective business rules. Other response transaction data fields will contain valid output.</td>
</tr>
<tr>
<td>FAILURE</td>
<td>The individual transaction failed processing according to business rules. Other response transaction data fields will not contain valid output.</td>
</tr>
<tr>
<td>WARNING</td>
<td>The individual transaction processed correctly but with warnings. Other response transaction data fields will contain valid output; however the Messages element will contain warning messages which should be investigated by the client in order to maintain data and processing integrity.</td>
</tr>
<tr>
<td>IGNORED</td>
<td>The individual transaction has been ignored. A transaction may be ignored if it matches various operation specific business rules.</td>
</tr>
</tbody>
</table>
2.8.6 Messages

As mentioned in the previous sections, both the RequestStatus and TransactionStatus fields contain the Messages element. This element is used to hold any messages concerned with either the request or its constituent transactions. The Messages element itself only contains an array of individual Message elements. Each Message element contains three fields: Code, Description and MessageSeverity. MessageSeverity can take on the following values:

<table>
<thead>
<tr>
<th>MessageSeverity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFORMATION</td>
<td>The message is of an informational nature.</td>
</tr>
<tr>
<td>WARNING</td>
<td>The message provides a warning which should be investigated. When a warning message is produced, it usually means that the request or transaction was successful, but with caveats.</td>
</tr>
<tr>
<td>ERROR</td>
<td>The message is produced when the request or transaction has failed. It contains reasons as to why it failed and what caused the error.</td>
</tr>
</tbody>
</table>
2.9 Error Handling

2.9.1 Schema Errors

Every request received by a HEIMS Applications and Offers Web Service is validated against the respective schema for that method. The validation performs checks to do with the format and structure of the request fields. If the request does not comply with the schema, the returning SOAP message will not contain any response data, only a SOAP fault. An example of a server response to an invalid request would look like:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:Body>
  <s:Fault>
    <faultcode>s:Client</faultcode>
    <faultstring xml:lang="en-AU">Xml Schema Validation Error(s)</faultstring>
    <detail>
        xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <ValidationErrors>
          <SchemaValidationFaultItem>
            <TransactionIdentifier>0</TransactionIdentifier>
            <ElementName>FamilyName</ElementName>
            <LineNumber>1</LineNumber>
            <ColumnNumber>1240</ColumnNumber>
            <Description>The 'http://dest.gov.au/Heims/:FamilyName' element is invalid - The value '' is invalid according to its datatype 'http://dest.gov.au/Heims/:FamilyName' - The actual length is less than the MinLength value.</Description>
          </SchemaValidationFaultItem>
        </ValidationErrors>
      </SchemaValidationFault>
    </detail>
  </s:Fault>
</s:Body>
</s:Envelope>
```

Within the `detail` element, HEIMS Applications and Offers Web Service will place a `SchemaValidationFault` element containing a `ValidationErrors` element. `ValidationErrors` element will contain a `SchemaValidationFaultItem` element for each error encountered. The `SchemaValidationFaultItem` element contains the following fields:

<table>
<thead>
<tr>
<th>SOAP Fault Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransactionIdentifier</td>
<td>This refers to the <code>RecordId</code> of the transaction data element that contains the error. This field applies to errors with transaction data only. Errors concerned with control data or errors to do with the entire request (e.g. duplicate <code>RecordId</code> values) will result in an empty <code>RecordId</code> value.</td>
</tr>
<tr>
<td>ElementName</td>
<td>The element in which the error occurred.</td>
</tr>
<tr>
<td>LineNumber</td>
<td>The line number of the incoming request where the error occurred.</td>
</tr>
<tr>
<td>ColumnNumber</td>
<td>The column number of the incoming request where the error occurred.</td>
</tr>
<tr>
<td>Description</td>
<td>A brief description of why the element was invalid.</td>
</tr>
</tbody>
</table>

Line and column number will only be useful if the tools used to call Web Services allow access to view the raw SOAP request sent over the wire. In any case, the other fields (`RecordId`, `Element` and `Description`) should be sufficient to pinpoint where the error occurred in the request.
Errors in the request control data (*RequestControlTable*) will return a SOAP fault immediately and no input transaction data will be validated. If any schema validation errors are detected, HEIMS Applications and Offers Web Service will not store the request which means the same *RequestId* can be re-used.

### 2.9.2 Business Errors

Once the incoming request has validated successfully against the schema, the server will process each transaction according to the respective business rules. Any errors or messages for a transaction data element arising from this validation are stored in the *TransactionStatus* field.
3 Interfaces

3.1 General Information
This section provides a list of all available Applications and Offers Web Services methods.

NOTE (1): It is necessary to use Basic Authentication via SSL to call any Applications and Offers web methods and the account used for calling the methods must be authorized to perform the business function implemented by the method.

End point URL:

WSDL Location

The available interface methods are:

<table>
<thead>
<tr>
<th>Section</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>LoadApplicationsAndOffersSubmission</td>
<td>To submit an Applications and Offers submission.</td>
</tr>
<tr>
<td>4.3</td>
<td>GetApplicationsAndOffersSubmissionResults</td>
<td>To retrieve the results of an Applications and Offers submission.</td>
</tr>
<tr>
<td>4.4</td>
<td>Ping</td>
<td>To test Web Service connectivity</td>
</tr>
</tbody>
</table>

3.2 LoadApplicationsAndOffersSubmission method
(for submission of Applications and Offers data for batch processing)

Method Signature
LoadSubmissionOutResponse LoadApplicationsAndOffersSubmission
(LoadApplicationsAndOffersSubmissionInRequest request)

Description
The LoadApplicationsAndOffersSubmission method can be used to load Applications and Offers data for new or updated submissions.

HEIMS Applications and Offers Web Service will validate the data against formatting rules. If the formatting rules are validated successfully, the submission will be placed in a batch queue. Any business rules defined in the Data Element Definitions (Appendix C) will be validated during the batch processing of the submission.

Requires Compression
Yes
3.3 **GetApplicationsAndOffersSubmissionResults method**  
(for retrieving the results of a processed submission of Applications and Offers data)

**Method Signature**
```
LoadSubmissionOutResponse GetApplicationsAndOffersSubmissionResults
  (GetSubmissionResultRequest request)
```

**Description**
This method retrieves the results of a submission that was previously submitted for batch processing.

**Requires Compression**
Yes

### 3.4 **Ping method**

**Method Signature**
```
string Ping()
```

**Description**
This method allows a client application to *ping* the server. It returns a string containing the date and time on the server.

**Requires Compression**
Yes
4 Security – Change Password

End point URL: https://app.heim.s.education.gov.au/Webservices.CHESSN/Security.asmx

Note: End Point URL is different for Change Password method, and does not use compression.

Method Signature

```
void ChangePassword(string logonId, string currentPassword, string newPassword)
```

Description

This method allows a user to change their password used to authenticate themselves on HEIMS web services. The following rules apply when changing passwords:

- The `logonId` given as the first parameter to the method must match the logon ID used to authenticate with HEIMS.
- The `currentPassword` must be valid for the logon ID.
- The `newPassword` must conform to the HEIMS password security rules.

Soap envelope for change password should look like:

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ChangePassword>
      <logonId>testuser</logonId>
      <currentPassword>Password1</currentPassword>
      <newPassword>Password2</newPassword>
    </ChangePassword>
  </soap:Body>
</soap:Envelope>
```

If any of the above rules are not met, a SOAP exception will be thrown. Within the SOAP fault, the `faultstring` tag will contain the error message. For example:

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>New Password must be between 7 and 15 characters long.</faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Once this method has been successfully called to change a password, it cannot be called again within a 24 hour period.

Requires Compression

No
5 Environments
There are two environments available to users of HEIMS Applications and Offers Web Services:

- Production; and
- Next Production Release (NPR).

The NPR environment is available to external departmental clients to enable them to develop or maintain their in-house systems that interface with the department's web services. Up-and-coming changes to the department's web services are made available in NPR as soon as is practicable.

As parent URL for both Production and Next Production Release environments are the same (app.heimss.education.gov.au), we recommend using End Point URLs mentioned in this document for each environment.

5.1 Production
See section 3 of this document for functionality available for HEIMS Applications and Offers Web Services live production environment.

The production end point is:

5.2 Next Production Release
An NPR environment will be made available to TACs to allow them to test functionality against the HEIMS Applications and Offers Web Services. Further details will be provided directly to TACs as they become available.

The NPR end point is:

6 Certificate
Application and Offers web service Certificate Authority (CA) has been changed from sanExternalSite.ssc.gov.au to san.education.gov.au (Baltimore CyberTrust Root). The update in certificate will encrypt all transmitted data between client machine and HEIMS web service server. The change in certificate has been implemented from July 2014. The sanExternalSite.ssc.gov.au certificate will be decommissioned on 27/02/2016. You will need to ensure that the latest certification is installed by 27/02/2016.

The san.education.gov.au (Baltimore CyberTrustRoot) certificate can be downloaded from https://app.heimss.education.gov.au/WCFServices/ApplicationsAndOffersSubmissionService.svc

Additional information to assist IT administrator with install the CA onto client system can be found at the following links.

- Exporting Certificate Authorities (CAs) from a Website
- How to call a Web service by using a client certificate for authentication in an ASP.NET Web application

7 Troubleshooting
If any problems are encountered with HEIMS Web Services, the first point of contact should be the HEIMS Helpdesk at HEIMS.datacollections@education.gov.au or phone on (02) 6240 7487.
A. Appendix A – References and Other Relevant Documents

A.1 References
The table below lists all external documents, web sites and other information sources that are referenced in this document or have been used in the gathering of requirements.

<table>
<thead>
<tr>
<th>Id</th>
<th>References</th>
</tr>
</thead>
</table>
| 1  | Web Services Architecture  
    W3C Working Group Note 11 February 2004  
    http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ |
| 2  | Extensible Markup Language (XML) 1.0 (Third Edition)  
    W3C Recommendation 04 February 2004  
    http://www.w3.org/TR/2004/REC-xml-20040204/ |
| 3  | XML Schema Part 1: Structures  
    W3C Recommendation 2 May 2001  
    http://www.w3.org/TR/xmlschema-1/ |
| 4  | XML Schema Part 2: DataTypes  
    W3C Recommendation 02 May 2001  
    http://www.w3.org/TR/xmlschema-2/ |
| 5  | SOAP Version 1.2 Part 1: Messaging Framework  
    W3C Recommendation 24 June 2003  
    http://www.w3.org/TR/2003/REC-soap12-part1-20030624/ |
| 6  | AS 4590-1999 Interchange of Client Information  
    Standards Australia |
| 7  | ISO/IEC 11179 Information Technology – Specification and Standardization of Data Elements |
| 8  | ISO/IEC 7064 Information Technology – Security techniques – Check character systems |

A.2 Other Relevant Documents
Other technical documentation may be sent to stakeholders to provide additional advice on integration with the HEIMS Applications and Offers Web Services if new issues or requirements arise.
## B. Appendix B – Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorisation</td>
<td>Determining a user’s access to a resource. Authorisation almost always relies on the user having been authenticated.</td>
</tr>
<tr>
<td>Authorised user</td>
<td>Authorised users are users authorised by the department.</td>
</tr>
<tr>
<td>Basic Authentication</td>
<td>A standard HTTP authentication protocol supported by most browsers where username and password is transmitted as Base-64 encoded text.</td>
</tr>
<tr>
<td>GUID</td>
<td>Globally Unique Identifier: A special type of identifier used in software applications in order to provide a reference number that is unique in the context for which it is used.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol: An application layer protocol that provides a standard for Web browsers and Web servers to communicate.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol (Secure): HTTP exchanged over an SSL encrypted session.</td>
</tr>
<tr>
<td>Megabyte</td>
<td>1,048,576 Bytes</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit: A software development kit (SDK) is typically a set of development tools that allows a software engineer to create applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar.</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer: A protocol developed by Netscape for transmitting private documents via the Internet. SSL works by using a private key to encrypt data that's transferred over the SSL connection. Both Netscape Navigator and Internet Explorer support SSL and many Web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with https: instead of http:.</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol: A suite of protocols that computers use to exchange information over the Internet.</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator A global identifier for a network-retrievable document.</td>
</tr>
<tr>
<td>UTF-8</td>
<td>8-bit UCS/Unicode Transformation Format A variable-length character encoding for Unicode. It is able to represent any character in the Unicode standard. UTF-8 is the standard encoding method for Email, web pages and XML documents.</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium: An international consortium of companies involved with developing standards for the Internet and the Web.</td>
</tr>
<tr>
<td>Web Services</td>
<td>Web Services are designed to support interoperable machine-to-machine interaction over a network through the use of open standards that enable systems built on different technologies to exchange data and supply services. The modern definition that Web Services always use the W3C Simple Object Access Protocol (SOAP) specification to format messages (regardless of transport) is used here.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| WSDL | Web Services Definition Language  
The standard format for describing a Web Service. Written in XML, it defines how to access a Web Service and what operations it will perform. |
| XML  | eXtensible Markup Language:  
A text based specification used to promote interoperable exchange of data through standardised validation mechanisms and expression of data in a human-readable, self-describing manner. |
| XSD  | XML Schema Definition  
XML schema, published as a W3C Recommendation in May 2001, can be used to express a scheme: a set of rules to which an XML document must conform in order to be considered ‘valid’ according to that schema. However, unlike most other schema languages, XML Schema was also designed with the intent of validation resulting in a collection of information adhering to specific data-types, which can be useful in the development of XML document processing software. |
C. Appendix C - Data Element Map

The Applications and Offers Data Element Map for the Tertiary Admissions Centres lists the data fields identified for the HEIMS Tertiary Admissions Centre submission process. This list provides a link between the business requirements in this document and the Applications and Offers Scope and Structure documents.

A summary description of the data elements is included on HEIMSHelp.

The rows of the Data Element Map are ordered by Data Grouping and by Data Element and includes the following columns.

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Meaning and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element Name</td>
<td>A unique name for each piece of data that is being collected.</td>
</tr>
<tr>
<td>Element Number</td>
<td>The element number in the Applications and Offers Submission Specifications.</td>
</tr>
<tr>
<td>Width</td>
<td>The maximum number of characters that each text field can contain.</td>
</tr>
<tr>
<td>Data Type</td>
<td>The type of the individual fields and information on the values that will be stored. Possible types are:</td>
</tr>
<tr>
<td></td>
<td>• Numeric – a number either integer or where required fractional.</td>
</tr>
<tr>
<td></td>
<td>• Text - a variable length text string.</td>
</tr>
<tr>
<td></td>
<td>• Date – a string representing a valid date.</td>
</tr>
<tr>
<td>Mandatory (Y or N)</td>
<td>Whether or not the individual field is mandatory or optional.</td>
</tr>
<tr>
<td></td>
<td>• Y - the data item is Mandatory, it must be provided before the set of information of which this field is a part can be successfully transferred into the department.</td>
</tr>
<tr>
<td></td>
<td>• N - the data item is Optional, it does not have to be provided before the set of information that this field is a part of can be successfully transferred into the department.</td>
</tr>
<tr>
<td></td>
<td>NOTE: Independent of whether or not a data item is mandatory or optional for the purpose of transferring the data into the department, other business rule cross checking that is applied during batch processing of the submission may require particular fields to contain particular values.</td>
</tr>
<tr>
<td>Parent Schema Entity</td>
<td>The XSD Complex Type that contains this data element.</td>
</tr>
<tr>
<td>Schema Element</td>
<td>The XSD Element in the Schema Entity that will be populated by this data element.</td>
</tr>
<tr>
<td>Element Name</td>
<td>Element Number</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Higher Education Provider code</td>
<td>306</td>
</tr>
<tr>
<td>Course of study type code</td>
<td>310</td>
</tr>
<tr>
<td>Date of birth</td>
<td>314</td>
</tr>
<tr>
<td>Gender code</td>
<td>315</td>
</tr>
<tr>
<td>Aboriginal and Torres Strait Islander code</td>
<td>316</td>
</tr>
<tr>
<td>Location code of permanent home residence</td>
<td>320</td>
</tr>
<tr>
<td>Basis for admission to offered course</td>
<td>327</td>
</tr>
<tr>
<td>Mode of attendance code</td>
<td>329</td>
</tr>
<tr>
<td>Year of arrival in Australia</td>
<td>347</td>
</tr>
<tr>
<td>Language spoken at home code</td>
<td>348</td>
</tr>
<tr>
<td>Citizen/resident indicator</td>
<td>358</td>
</tr>
<tr>
<td>Tertiary Entrance Score</td>
<td>369</td>
</tr>
<tr>
<td>Surname</td>
<td>402</td>
</tr>
<tr>
<td>Given name – first</td>
<td>403</td>
</tr>
<tr>
<td>Given names – other</td>
<td>404</td>
</tr>
<tr>
<td>Postal address – part 1</td>
<td>406</td>
</tr>
<tr>
<td>Postal address – part 2</td>
<td>407</td>
</tr>
<tr>
<td>Postal address – postcode</td>
<td>409</td>
</tr>
<tr>
<td>Element Name</td>
<td>Element Number</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Field of education code</td>
<td>461</td>
</tr>
<tr>
<td>Field of education supplementary code</td>
<td>462</td>
</tr>
<tr>
<td>Postal address – suburb/town</td>
<td>466</td>
</tr>
<tr>
<td>Postal address – state code</td>
<td>467</td>
</tr>
<tr>
<td>Postal address – country name</td>
<td>468</td>
</tr>
<tr>
<td>Highest educational participation</td>
<td>493</td>
</tr>
<tr>
<td>Course campus postcode</td>
<td>559</td>
</tr>
<tr>
<td>Application identification code</td>
<td>700</td>
</tr>
<tr>
<td>Application status</td>
<td>701</td>
</tr>
<tr>
<td>Australian Year 12 result type code</td>
<td>702</td>
</tr>
<tr>
<td>Element Name</td>
<td>Element Number</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>Date offered</td>
<td>705</td>
</tr>
</tbody>
</table>
| Intake date                                    | 709            | 6     | Numeric    | Y           | Preference & Offer   | IntakeDate                                   | Year: 2009-2099  
 |                                                |                |       |            |              |                                                   | Month: 00-12    |
| International Baccalaureate aggregate score    | 710            | 2     | Text      | N           | AoApplication       | InternationalBaccalaureateAggregateScore      | 21-45, 99    |
| Interstate Transfer Index                      | 711            | 5     | Text      | N           | AoApplication       | InterstateTransferIndex                      | 00.00-99.95  |
| Preference ordinal position                    | 713            | 2     | Text      | Y           | Preference & Offer  | PreferenceOrdinalPosition                    | 00-99       |
| Response code                                  | 723            | 1     | Text      | N           | Offer                | ResponseCode                                 | 1, 2, 3, 4, 5, 6 |
| Sector code                                    | 724            | 1     | Text      | Y           | Preference & Offer  | SectorCode                                   | 1, 2, 3, 4   |
| Postal address – country code                  | 728            | 4     | Text      | N           | AoAddress            | CountryCode                                  | 1101-9299   |
| Overall Position                               | 729            | 2     | Text      | N           | AoApplication        | OverallPosition                              | 01-25, 99   |
| Prior postgraduate course; year                | 730            | 5     | Text      | N           | AoApplication        | PriorPostGraduateCourseCodeYear              | Prior Postgraduate Course Code Year  
 |                                                |                |       |            |              |                                                   | First digit: 1, 2, 3  
 |                                                |                |       |            |              |                                                   | If 1st digit is 1, last 4 must be 0000, otherwise last 4 must be in range 1900-2099, or 9999 |
| Prior bachelor course; year                    | 731            | 5     | Text      | N           | AoApplication        | PriorBachelorDegreeCourseCodeYear            | Prior Bachelor Degree Course Code Year  
 |                                                |                |       |            |              |                                                   | *           |
| Prior sub-degree course (HE); year            | 732            | 5     | Text      | N           | AoApplication        | PriorHeSubDegreeCourseCodeYear               | Prior Sub-degree Course Code Year  
 |                                                |                |       |            |              |                                                   | *           |
| Prior sub-degree course (VET); year           | 733            | 5     | Text      | N           | AoApplication        | PriorVetSubDegreeCourseCodeYear              | Prior Sub-degree Course Code Year  
 |                                                |                |       |            |              |                                                   | *           |
| Prior VET award course; year                   | 734            | 5     | Text      | N           | AoApplication        | PriorVetAwardCourseCodeYear                  | Prior VET Award Course Code Year  
 |                                                |                |       |            |              |                                                   | *           |
| Prior secondary education course through VET sector; year | 735        | 5     | Text      | N           | AoApplication        | PriorVetSecondaryEducationCourseCodeYear     | Prior Secondary Education Course Code Year  
 |                                                |                |       |            |              |                                                   | First digit: 1, 2  
 |                                                |                |       |            |              |                                                   | If 1st digit is 1, last 4 must be 0000, otherwise last 4 must be in range 1900-2099, or 9999 |
| Prior secondary education course at school; year | 736        | 5     | Text      | N           | AoApplication        | PriorSchoolSecondaryEducationCourseCodeYear  | Prior Secondary Education Course Code Year  
<p>|                                                |                |       |            |              |                                                   | *           |</p>
<table>
<thead>
<tr>
<th>Element Name</th>
<th>Element Number</th>
<th>Width</th>
<th>Data type</th>
<th>Mandatory?</th>
<th>Parent Schema Entity</th>
<th>Schema Element Name</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior other qualification/certificate; year</td>
<td>737</td>
<td>5</td>
<td>Text</td>
<td>N</td>
<td>AoApplication</td>
<td>PriorOtherCertificateQualificationCourseCodeYear</td>
<td></td>
</tr>
<tr>
<td>Request ID</td>
<td>36</td>
<td></td>
<td>Text</td>
<td>Y</td>
<td>RequestControl Table</td>
<td>RequestId</td>
<td></td>
</tr>
<tr>
<td>Client Organisation Code</td>
<td>4</td>
<td></td>
<td>Text</td>
<td>Y</td>
<td>RequestControl Table</td>
<td>ClientOrganisationCode</td>
<td>Must match the client organisation code of the logged on user</td>
</tr>
<tr>
<td>Request Local Date Time</td>
<td></td>
<td></td>
<td>Date-Time</td>
<td>Y</td>
<td>RequestControl Table</td>
<td>RequestLocalDateTime</td>
<td>YYYY-MM-DDTHH:mm:SS.s +zzzz See <a href="http://www.w3.org/TR/xmlschema-2/#dateTime">http://www.w3.org/TR/xmlschema-2/#dateTime</a></td>
</tr>
<tr>
<td>Submission Reference Year</td>
<td>4</td>
<td></td>
<td>Text</td>
<td>Y</td>
<td>SubmissionMetadata</td>
<td>SubmissionReferenceYear</td>
<td>Current year or next year</td>
</tr>
<tr>
<td>Submission Reporting Period</td>
<td>1</td>
<td></td>
<td>Text</td>
<td>Y</td>
<td>SubmissionMetadata</td>
<td>SubmissionReportingPeriod</td>
<td>1-5</td>
</tr>
<tr>
<td>Submission Number</td>
<td>1</td>
<td></td>
<td>Numeric</td>
<td>Y</td>
<td>SubmissionMetadata</td>
<td>SubmissionNumber</td>
<td></td>
</tr>
<tr>
<td>Commonwealth Higher Education Student Support Number (CHESSN)</td>
<td>488</td>
<td>10</td>
<td>Text</td>
<td>N</td>
<td>AoApplication</td>
<td>CHESSN</td>
<td>E488 (CHESSN) if NOT NULL or ZZZZZZZZZZZZZ must be a valid string:(1000000000 to 9999999999)</td>
</tr>
</tbody>
</table>

* Optional elements that are blank or contain only spaces should not be passed in the XML
D. Applications and Offers Web Service Requests and Responses

D.1 LoadApplicationsAndOffersSubmission

```xml
<xs:element name="LoadApplicationsAndOffersSubmissionInRequest">
  <xs:complexType>
    <xs:annotation>
      <xs:documentation>A request to the LoadOffersSubmission method.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SubmissionMetaData" type="SubmissionMetadata" minOccurs="1" maxOccurs="1"/>
      <xs:element name="TransactionData" type="ArrayOfLoadApplicationsAndOffersSubmissionIn" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="LoadSubmissionResponse">
  <xs:annotation>
    <xs:documentation>A response from Load Submission calls.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ResponseControlData" type="ResponseControlTable" minOccurs="1" maxOccurs="1"/>
      <xs:element name="LoadSubmissionResults" type="ArrayOfLoadSubmissionOut" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```
D.2 GetApplicationsAndOffersSubmissionResultsRequest

<xsd:element name="GetApplicationsAndOffersSubmissionResultsRequest">
  <xsd:annotation>
    <xsd:documentation>Returns submission results for a request id.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1" />
      <xsd:element name="SubmissionRequestId" type="RequestId" minOccurs="1" maxOccurs="1" />    
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="GetApplicationsAndOffersSubmissionResultsResponse">
  <xsd:annotation>
    <xsd:documentation>Returns submission results.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="ResponseControlData" type="ResponseControlTable" minOccurs="1" maxOccurs="1" />
      <xsd:element name="GetApplicationsAndOffersSubmissionResults" type="ArrayOfLoadSubmissionOut" minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>