HIGHER EDUCATION INFORMATION MANAGEMENT SYSTEM (HEIMS)

HEIMS WEB SERVICES INTERFACE

TECHNICAL SPECIFICATION
COMMONWEALTH SCHOLARSHIP 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

2 Stakeholders and Users ........................................................................................................ 4
   2.1 Stakeholder Responsibilities in Relation to This Document ............................................ 4

3 HEIMS Technical Architecture ............................................................................................... 5
   3.1 Web Services Architecture Overview .............................................................................. 5
   3.2 Technical Requirements ................................................................................................. 6
   3.3 Compression ................................................................................................................... 6
   3.4 Security .......................................................................................................................... 7
   3.5 Transactions Overview .................................................................................................... 8
      3.5.1 Real-Time Transactions ............................................................................................. 8
      3.5.2 Batch Transactions .................................................................................................... 9
   3.6 Service Agreements and Availability ............................................................................ 10
   3.7 XML Schemas ................................................................................................................ 11
      3.7.1 LoadScholarshipSubmissionIn ................................................................................. 11
      3.7.2 LoadSubmissionOut ................................................................................................. 12
      3.7.3 SearchScholarshipSubmissionIn ............................................................................. 13
      3.7.4 SearchSubmissionOut .............................................................................................. 14
      3.7.5 SearchScholarshipIn ................................................................................................. 15
      3.7.6 SearchScholarshipOut ............................................................................................... 16
      3.7.7 RequestControlTable ............................................................................................... 17
      3.7.8 ResponseControlTable ............................................................................................ 18
   3.8 Request and Response Definitions .................................................................................. 19
      3.8.1 Request Identifiers ..................................................................................................... 19
      3.8.2 Request Object Schema ........................................................................................... 20
      3.8.3 Response Schema .................................................................................................... 21
      3.8.4 Control Data ............................................................................................................. 22
      3.8.5 Transaction Data ....................................................................................................... 23
      3.8.6 Messages .................................................................................................................. 23
   3.9 Error Handling ................................................................................................................ 24
      3.9.1 Schema Errors .......................................................................................................... 24
      3.9.2 Business Errors ........................................................................................................ 24

4 Interfaces ............................................................................................................................... 25
   4.1 General Information ......................................................................................................... 25
      LoadScholarshipSubmission – Real Time and Batch submission of scholarship data .......... 26
   4.2 GetSubmissionStatus ....................................................................................................... 27
   4.3 GetScholarshipTransactionResult .................................................................................. 28
   4.4 SearchScholarshipSubmission ......................................................................................... 29
   4.5 SearchScholarshipData .................................................................................................... 30
   4.6 Ping ................................................................................................................................. 31

5 Security – Change Password ................................................................................................. 32

6 Environments ......................................................................................................................... 33
   6.1 Production ....................................................................................................................... 33

7 Certificate .............................................................................................................................. 33

8 Troubleshooting ................................................................................................................... 34

A. Appendix A – References and Other Relevant Documents ................................................. 35
   A.1 References ..................................................................................................................... 35
   A.2 Other Relevant Documents ........................................................................................... 35

B. Appendix B – Glossary .......................................................................................................... 36

C. Data Element Definitions ..................................................................................................... 38

D. Web Service Schemas .......................................................................................................... 43
   D.1 BaseTypes.xsd .............................................................................................................. 43
   D.2 ComplexTypes.xsd ....................................................................................................... 53
   D.3 ScholarshipWebServiceRequestAndResponse.xsd ...................................................... 58
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 Higher Education Providers (HEPs) systems and the Higher Education Information Management System (HEIMS). The scope of the document includes technical specifications of the Web Services for Scholarship Submissions, description of the web methods, the schemas used and troubleshooting information.

Due to the flexibility of Web Services and the large number of systems they can run on, 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 scholarship submission data;
- Technical specification for any of the W3C standards used in HEIMS; and
- Data submission and retrieval for other HEIMS data, eg. student details.

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 Higher Education Providers (HEP Developers and HEIMS System Implementers).

The Scholarships Web Services described in this document are for use by Higher Education Providers (HEPs) only. They are not for use by Tertiary Admission Centres (TACs) or Vocational Education and Training (VET) Providers.

1.4 Disclaimer
These specifications give information about how to use the HEIMS Scholarships 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 HEPS and others using these specifications to ensure they are using the latest version.
2 Stakeholders and Users

2.1 Stakeholder Responsibilities in Relation to This Document
It is the responsibility of the stakeholders to review this document and ensure all required information is present. Stakeholders can raise any matters relating to this document through HEIMS Data Collections at: HEIMS.datacollections@Education.gov.au.
3 HEIMS Technical Architecture

3.1 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.

3.2 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).

From 01 October 2018 onwards, communications through transport methods SSL3.0, TLS1.0 and TLS1.1 is no longer supported.

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; and
- 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 Scholarship 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.
3.3 Compression

Compression is a standard feature in HEIMS Scholarship Web Services. All HEIMS Scholarship 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 HEIMS Web Services is gzip.

In order to use compression, the tools used to call HEIMS Scholarship 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:

  ```
  <Compression s:mustUnderstand="1" Algorithm="http://dest.gov.au/Heims/compression/gzip"
  ```

<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:

  ```
  ```

For example: (Dummy Data used in the example)

Before Compression:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <To s:mustUnderstand="1" xmlns="http://schemas.microsoft.com/ws/2005/05/addressing/none">http://localhost:2350/HeimsScholarshipSubmissionService.svc</To>
  </s:Header>
  <s:Body>
    <GetScholarshipSubmissions xmlns="http://dest.gov.au/Heims/Service">
        <d4p1:RequestControlData>
          <d4p1:RequestId>7ce69824-668b-a064-38dda2fa06c9</d4p1:RequestId>
          <d4p1:ClientOrganisationCode>0000</d4p1:ClientOrganisationCode>
          <d4p1:RequestLocalDateTime>2007-12-12T09:29:02.5473372+11:00</d4p1:RequestLocalDateTime>
        </d4p1:RequestControlData>
        <d4p1:GetScholarshipSubmissionIn>
          <d4p1:ClientRequestId>5ce69824-668b-a064-38dda2fa06c8</d4p1:ClientRequestId>
        </d4p1:GetScholarshipSubmissionIn>
      </request>
    </GetScholarshipSubmissions>
  </s:Body>
</s:Envelope>
```

After Compression:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Header>
    <Compression s:mustUnderstand="1" Algorithm="http://dest.gov.au/Heims/compression/gzip"
  </s:Header>
  <s:Body>
    <GetScholarshipSubmissions xmlns="http://dest.gov.au/Heims/Service">
        <d4p1:RequestControlData>
          <d4p1:RequestId>7ce69824-668b-46a8-a064-38dda2fa06c9</d4p1:RequestId>
          <d4p1:ClientOrganisationCode>0000</d4p1:ClientOrganisationCode>
          <d4p1:RequestLocalDateTime>2007-12-12T09:29:02.5473372+11:00</d4p1:RequestLocalDateTime>
        </d4p1:RequestControlData>
        <d4p1:GetScholarshipSubmissionIn>
          <d4p1:ClientRequestId>5ce69824-668b-46a8-a064-38dda2fa06c8</d4p1:ClientRequestId>
        </d4p1:GetScholarshipSubmissionIn>
      </request>
    </GetScholarshipSubmissions>
  </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>` 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" Algorithm="http://dest.gov.au/Heims/compression/gzip"
```

<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
```

The client will have to intercept the response from the Server and will have to uncompress the stream with in the `<SC:CompressedData>` into the original SOAP message body and process it as normal.

### 3.4 Security

HEIMS Scholarship 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 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 IT Liaison officer (HEIMS.datacollections@Education.gov.au) 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.

Change of password can be done via the `ChangePassword` web method described in section 6.

### 3.5 Transactions Overview

HEIMS Scholarship provides two types of Web Services implementations: real-time and batch. The main difference is that real-time Web Services are processed immediately by the server, whereas batch Web Services are queued on the server to be processed later. The `LoadScholarshipSubmission` method has a real-time and a batch implementation. Both versions perform exactly the same business function. The total number of transactions in a request is used to determine whether a request will be submitted as real-time or batch. Currently the maximum real time request is set to 100. Any `LoadScholarshipSubmission` request containing more than 100 transactions will be placed in the batch queue on the server to be processed later.

#### 3.5.1 Real-Time Transactions

Real-time transactions are the simpler of the two types of transactions. A real-time method will perform the requested action on the server immediately and return the results as part of the call. Real-time requests can only accept up to 100 transactions for processing at a time and should only be used when a result is required immediately.

![Figure 1: Message Flow in Real-Time Transactions](image-url)
3.5.2 Batch Transactions

Batch transactions are those that, due to the volume of data being sent or the amount of processing required, will be placed in a queue for later processing. They are used primarily where an immediate response is not required. Batch methods can contain requests for multiple transactions and therefore can be very large in size.

An invocation of a batch transaction can actually comprise of three 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 if processing for this request is complete. The third and final call queries the server 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 2: Message Flow in Batch Transactions
3.6 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.
- **Real Time transactions** – 95% of transactions will be processed within 60 seconds 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 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 Scholarship 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 Scholarship Web Services will generally be available during this period but interruptions to server 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.
3.7 XML Schemas
This section describes the top level XML Schema complex types used in Scholarship Web Service Request and Responses.

The actual request and response types are described in section 4.8. This section will be used to show the high-level business objects that can be modified directly through Scholarship Web Services. Full schema definitions for all HEIMS data types can be found in Appendix D. Types referenced in this section that are not represented by complex types are actually simple types.

Note that, in the class diagrams below, Association Roles (adorning the arrowheads) are represented as elements in the XSD Schemas. 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. Where not specified, the default multiplicity is always one.

Appendix D.1: XML Schema Simple Types lists the simple types and restrictions (defined in the XML Schema) that are reused by many of the complex types.

3.7.1 LoadScholarshipSubmissionIn
LoadScholarshipSubmissionIn refers to transaction data that contains student information plus details of the scholarship the student is currently entitled to receive.

LoadScholarshipSubmissionIn is part of the LoadScholarshipSubmissionInRequest

Note:
Data element VariationReasonCode in StudentScholarshipDetails indicates the purpose of the student scholarship record.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VariationReasonCode</td>
<td>xs:string</td>
<td>A 1-digit code representing the purpose of the student scholarship record.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values 0, 1 or 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 – First time a record is reported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 – Modify a previously reported record</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 – Delete a previously reported record</td>
</tr>
</tbody>
</table>
3.7.2 LoadSubmissionOut

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

LoadSubmissionOut is part of the LoadScholarshipSubmissionResponse.

Figure 4: LoadSubmissionOut XSD complex type

Note: Input Stream is a base64binary stream. Input Stream contains input transaction data of type LoadScholarshipSubmissionIn which is used for providing student details plus scholarship details. Refer to section 4.7.1 for more information on LoadScholarshipSubmissionIn.
3.7.3 SearchScholarshipSubmissionIn

SearchScholarshipSubmissionIn contains the search criteria for retrieving scholarship submission details.

SearchScholarshipSubmissionIn is part of SearchScholarshipSubmissionRequest

Note: Only Scholarship submissions done after February 2008 are available for searching.

Figure 5: ScholarshipSubmissionIn XSD complex type
### 3.7.4 SearchSubmissionOut

`SearchSubmissionOut` contains the details of Scholarship submission.

SearchSubmissionOut is part of `SearchScholarshipSubmissionResponse`

![Figure 6: SearchSubmissionOut XSD complex type](image-url)

**Class Diagram**

- `<xsd:complexType>`
  - `<xsd:element>`
    - `createDateTime: dateTime`
    - `createLogonName: string`
    - `loadCount: int`
    - `loadStatus: string`
    - `loadType: string`
    - `submissionRevisionNumber: int`
    - `submissionYear: gYear`
    - `updateDateTime: dateTime`
    - `updateLogonName: string`

- `<xsd:attribute>`
  - `clientOrganisationCode: string`
3.7.5 SearchScholarshipIn

SearchScholarshipIn contains the search criteria for retrieving student’s scholarship details from HEIMS Database.

SearchScholarshipIn is part of SeachScholarshipDataRequest

```
<xs:complexType name="SearchScholarshipIn">
  <xs:sequence>
    <xs:element name="ScholarshipCode" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="ScholarshipStatusCode" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="ScholarshipTerminationReasonCode" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="StudentId" type="xs:int" minOccurs="0" maxOccurs="1"/>
    <xs:element name="StartPageNumber" type="xs:int" minOccurs="0" maxOccurs="1"/>
    <xs:element name="RecordStatus" type="ScholarshipRecordStatus[0..1]" minOccurs="0" maxOccurs="1"/>
    <xs:element name="ReportingPeriod" type="ReportingPeriod[0..1]" minOccurs="0" maxOccurs="1"/>
    <xs:element name="ReportingYear" type="xs:string" minOccurs="0" maxOccurs="1"/>
    <xs:element name="ReturnToSender" type="ReturnToSenderFlag[0..1]" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
</xs:complexType>
```

Figure 7: SearchScholarshipIn XSD complex type

Note:
SearchScholarshipData method returns paged results. StartPageNumber data element in SearchScholarshipIn provides the ability to request a specific page. For the first request StartPageNumber should be 1.

Refer SearchScholarshipOut in Section 4.7.6 for PageSize, TotalPageCount and TotalCount data elements definitions.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartPageNumber</td>
<td>xs:int</td>
<td>Specifies the start page in the range of expected results pages. For the first request this must be 1.</td>
</tr>
</tbody>
</table>
3.7.6 SearchScholarshipOut

SearchScholarshipOut contains the scholarship details of the students.

SearchScholarshipOut is part of the SearchScholarshipDataResponse.

Figure 8: SearchScholarshipOut XSD complex type

Note:
SearchScholarshipData method returns paged results. SearchScholarshipOut contains data elements which contain information associated with paging.

Following data elements are used in paging and is part of SearchScholarshipDataResponse:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Size</td>
<td>xs:int</td>
<td>Specifies the number of records in each page.</td>
</tr>
<tr>
<td>TotalPageCount</td>
<td>xs:int</td>
<td>Total Number of pages returned</td>
</tr>
<tr>
<td>TotalCount</td>
<td>xs:int</td>
<td>Total records found</td>
</tr>
</tbody>
</table>

Refer SearchScholarshipIn in Section 4.7.5 for StartPageNumber data element definition.

Following data elements provide additional information about student scholarship:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScholarshipDeleteFlag</td>
<td>xs:boolean</td>
<td>Specifies whether a scholarship is deleted or not</td>
</tr>
<tr>
<td>ScholarshipInvalidFlag</td>
<td>xs:boolean</td>
<td>Specifies whether a scholarship is invalid or not</td>
</tr>
<tr>
<td>ScholarshipUpdateDateTime</td>
<td>xs:date</td>
<td>DateTime when a scholarship was updated</td>
</tr>
<tr>
<td>ScholarshipCreateDateTime</td>
<td>xs:date</td>
<td>DateTime when a scholarship was created</td>
</tr>
</tbody>
</table>
3.7.7 RequestControlTable

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

Figure 9: RequestControlTable XSD complex type
3.7.8 ResponseControlTable

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

![ResponseControlTable XSD complex type](image)

**Figure 10: ResponseControlTable XSD complex type**
3.8 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 Appendix D.

3.8.1 Request Identifiers

Request identifiers (RequestIds) are an important part of HEIMS Scholarship 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 format of the RequestId must be "HHHHCS YYYYMMDD HH:MM:SS:NN" where:

- "HHHH" is your Client Organisation Code
- "CS" is the literal text "CS"
- "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 HEP and it must retain uniqueness over time. It is the responsibility of the HEP 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.
3.8.2 Request Object Schema

The request object is a generic container that holds all the information a client needs to send to a HEIMS Scholarship Web Service. It contains request control data (one RequestControlTable) plus multiple request transaction data elements. If the request is being sent for a real-time method, only 100 transaction data elements are allowed. For batch methods, there is no limit on the number of transactions allowed per request.

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

![Figure 11: General Request Class Diagram](image)

The classes in the Figure 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.
3.8.3 Response Schema

Similarly, the response schema is a container for all information a Scholarship Web Service sends to a client. It contains response control data (one ResponseControlTable) plus multiple response transaction data elements.

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

![General Response Class Diagram](image)

**Figure 12: General Response Class Diagram**

The classes in Figure 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.
3.8.4 Control Data

Control data is present in all communication between clients and HEIMS Scholarship Web Services. It provides information concerned with the HEIMS Scholarship Web Service infrastructure and as such, is essential to every HEIMS Scholarship 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 Scholarship Web Service, a RequestControlTable must always be sent. Depending on the type of method being called (real-time or batch), it can be sent by itself or 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 Scholarship Web Service, a ResponseControlTable is always present. Again, depending on the method being called (real-time or batch), it can be sent by itself or 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 RequestStatusCode – the current status of the request, and Messages – 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 RequestId 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 RequestId 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.
3.8.5 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.

TransactionData is currently being used in LoadScholarshipSubmissionRequest only. SubmissionResults (Response TransactionData) is only used in DataSubmissionResponse only.

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 HEP 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>

3.8.6 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>
3.9 Error Handling

3.9.1 Schema Errors
Every request received by a HEIMS Scholarship 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:

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <s:Fault>
      <faultcode>s:Client</faultcode>
      <faultstring>Xml Schema Validation Error(s)</faultstring>
      <detail>
          <ValidationErrors>
            <SchemaValidationFaultItem>
              <TransactionIdentifier>1</TransactionIdentifier>
              <ElementName>GivenName</ElementName>
              <LineNumber>1</LineNumber>
              <ColumnNumber>667</ColumnNumber>
              <Description>The 'http://dest.gov.au/Heims/:GivenName' element is invalid - The value 'Very_Very_Very_Very_Very_Very_Very_Very_long_GivenName' is invalid according to its datatype 'http://dest.gov.au/Heims/:GivenName' - The actual length is greater than the MaxLength value.</Description>
            </SchemaValidationFaultItem>
          </ValidationErrors>
        </SchemaValidationFault>
        <ValidationErrors>
          <SchemaValidationFaultItem>
            <TransactionIdentifier>1</TransactionIdentifier>
            <ElementName>GivenName</ElementName>
            <LineNumber>1</LineNumber>
            <ColumnNumber>667</ColumnNumber>
            <Description>The 'http://dest.gov.au/Heims/:GivenName' element is invalid - The value 'Very_Very_Very_Very_Very_Very_Very_Very_long_GivenName' is invalid according to its datatype 'http://dest.gov.au/Heims/:GivenName' - The actual length is greater than the MaxLength value.</Description>
          </SchemaValidationFaultItem>
        </ValidationErrors>
      </detail>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

Within the `detail` element, HEIMS Scholarship 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 (eg. 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 Scholarship Web Service will not store the request which means the same RequestId can be re-used.

3.9.2 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.
4 Interfaces

4.1 General Information

This section provides a list of all available scholarship Web Services and the methods associated with them. It also lists data types used and relevant business rules.

![Figure 13: Web Service Methods](image)

It is necessary to use Basic Authentication via SSL to call these methods.

**End point URL:** https://app.heim.education.gov.au/WCFServices/ScholarshipSubmissionService.svc

**WSDL Location:** https://app.heim.education.gov.au/WCFServices/ScholarshipSubmissionService.svc?wsdl

The interface methods are described in the subsection that follows:

<table>
<thead>
<tr>
<th>Section</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>LoadScholarshipSubmission:</td>
<td>To load Student Scholarship details.</td>
</tr>
<tr>
<td>5.3</td>
<td>GetSubmissionStatus:</td>
<td>To retrieve Status of scholarship submission.</td>
</tr>
<tr>
<td>5.4</td>
<td>GetScholarshipTransactionResult:</td>
<td>To retrieve transactions results of scholarships submissions.</td>
</tr>
<tr>
<td>5.5</td>
<td>SearchScholarshipSubmission:</td>
<td>To search Scholarship submission.</td>
</tr>
<tr>
<td>5.6</td>
<td>SearchScholarshipData:</td>
<td>To search Scholarship data.</td>
</tr>
<tr>
<td>5.7</td>
<td>Ping</td>
<td>To test Web Service connectivity.</td>
</tr>
</tbody>
</table>

A HEP can only modify its own submission data. The account issued for calling these Web Services will be able to perform all of the above actions.
4.2 LoadScholarshipSubmission – Real Time and Batch submission of scholarship data

Figure 14: LoadScholarshipSubmission

Method Signature

LoadSubmissionOutResponse LoadScholarshipSubmission (LoadScholarshipSubmissionInRequest request)

Description

LoadScholarshipSubmission can be used to load scholarship data for new or continuing students. Based on the number of transactions in the submission real time or batch processing of the submission will be done. For real time submissions (i.e. if the number of transactions in the submission is less than or equal to the real time transaction limit), request will be processed immediately and response object will be returned immediately.

Records within a submission will be processed in the order they are received. Therefore records must be sent in the correct order. For example, if a submission contains an Add transaction (VariationReasonCode=0) and a Modify transaction (VariationReasonCode=1) for the same scholarship record and that record does not yet exist in HEIMS, then the Add transaction must come first in the submission. If the Modify transaction comes first then the Web Service will return an error because it won’t be able to find the record to modify.

HEIMS Scholarship Web Service will validate the data against formatting and business rules defined in the Data Element Definitions (Appendix C)

Requires Compression

Yes
4.3 GetSubmissionStatus

Figure 15: GetSubmissionStatus

Method Signature
GetSubmissionStatusResponse GetSubmissionStatus (GetSubmissionStatusRequest request)

Description
GetSubmissionStatus can be used for retrieving the status of a scholarship submission.

HEIMS Scholarship Web Service will validate the data against formatting and business rules defined in the Data Element Definitions (Appendix C)

Requires Compression
Yes
4.4 GetScholarshipTransactionResult

Method Signature

GetScholarshipTransactionResultResponse GetScholarshipTransactionResult
(GetScholarshipTransactionResultRequest scholarshipResultRequest)

Description

This method can be used for retrieving Transaction Results for one or more scholarship submissions. This method also provides an option to filter the results based on the TransactionStatusCode. If TransactionStatusCode is Null or empty all results will be returned.

HEIMS Scholarship Web Service will validate the data against formatting and business rules defined in the Data Element Definitions (Appendix C)

Requires Compression

Yes

Note:

SubmissionResults in GetScholarshipTransactionResultResult is currently configured to return a maximum of 10,000 LoadSubmissionOut structures.
4.5 **SearchScholarshipSubmission**

Method Signature:
```
SearchScholarshipSubmissionsResponse SearchScholarshipSubmission (SearchScholarshipSubmissionRequest controlRequest)
```

**Description**

This method can be used to search and retrieve Scholarship submissions.

HEIMS Scholarship Web Service will validate the data against formatting and business rules defined in the Data Element Definitions (Appendix C)

Note: This method will only return Scholarship submissions done after February 2008.

**Requires Compression**

Yes
4.6 SearchScholarshipData

**Figure 18: SearchScholarshipData**

**Method Signature**
SearchScholarshipDataResponse SearchScholarshipData (SearchScholarshipDataRequest controlRequest)

**Description**
This method can be used to search and retrieve Scholarship Data.

HEIMS Scholarship Web Service will validate the data against formatting and business rules defined in the Data Element Definitions (Appendix C)

**Requires Compression**
Yes
4.7 Ping

Method Signature

\textit{string Ping} ()

Description

This method allows a client application to \textit{ping} the server. It returns a string containing the date and time on the server.

Requires Compression

Yes
### 5 Security – Change Password


Note: End Point URL is different for Change Password method.

**Method Signature**

```csharp
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>
```

Refer to section 4.4 for information related to security and passwords. 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>
```

**Requires Compression**

No
6 Environments

6.1 Production

See section 5 of this document for functionality available for HEIMS Scholarship Web Services live production environment.

7 Certificate

The Commonwealth Scholarship 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.veis.edu.au/WCFServices/ScholarshipSubmissionService.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
8 Troubleshooting

If any problems are encountered with HEIMS Web Services, the first point of contact should be the HEIMS Developers Network. Located on the Developer Network is a list of frequently asked questions and a message board to participate in technical discussions.

For further technical enquiries, please contact 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
This sub-section 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 Scholarship 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 include Panel Members, Advisors and other users authorised by the department. This will include authorised department officers during the Assessment Period. Additional Commonwealth officers may be authorised to use the HEIMS Scholarship Web Service following the Assessment Period.</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>Term</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| UTF-8           | 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. |
| W3C             | World Wide Web Consortium:  
An international consortium of companies involved with developing standards for the Internet and the Web.                                                                                                                                                                   |
| Web Services    | 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. |
| 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. Data Element Definitions

The Data Dictionary lists the data fields identified for the HEIMS Scholarship Submission process. This list provides a link between the business requirements in the Scholarship Submission Specifications and the Scholarship Technical Specifications as documented in the Scholarship Technical Specifications Documentation Package.

A summary description of the data elements is included in the Data Dictionary. For the complete definition of the terms and the context of the requirements, refer to the Scholarship Submission Specifications.

The rows of the data dictionary are ordered by Data Grouping and by Data Element.

The Data Dictionary contains the following columns:

<table>
<thead>
<tr>
<th>Column Heading</th>
<th>Meaning and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Grouping</td>
<td>A column that groups fields together to provide a particular set of information. For example, the Student Details data grouping contains fields that describe details about a student.</td>
</tr>
<tr>
<td>Data Element</td>
<td>A unique business name for each piece of data that is being collected.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of the individual fields and information on the values that will be stored. The possible types are:</td>
</tr>
<tr>
<td></td>
<td>• Boolean – true or false.</td>
</tr>
<tr>
<td></td>
<td>• Number – a number either integer or where required fractional.</td>
</tr>
<tr>
<td></td>
<td>• Text - a variable length text string.</td>
</tr>
<tr>
<td>Length</td>
<td>The maximum number of characters that each text field can contain.</td>
</tr>
<tr>
<td>Mandatory (M) or Optional (O)</td>
<td>Whether or not the individual field is mandatory or optional.</td>
</tr>
<tr>
<td></td>
<td>• M - 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>• O - 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>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 or after the submission may require particular fields to contain particular values.</td>
</tr>
<tr>
<td>Submission Spec Reference</td>
<td>The section number in the Scholarship Submission Specifications that refers to the business rule the data item supports or references.</td>
</tr>
<tr>
<td>Description</td>
<td>A general description of the data element, what it means and where appropriate what values it may be given.</td>
</tr>
<tr>
<td>XML Schema Type</td>
<td>A cross reference into the XML Schemas giving the XSD Complex Type that this data element belongs to.</td>
</tr>
<tr>
<td>XML Schema Element</td>
<td>A cross reference into the XML Schemas giving the XML Element that this data element is named.</td>
</tr>
<tr>
<td>Data Grouping</td>
<td>Data Element</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Student Group</td>
<td>Student identification code</td>
</tr>
<tr>
<td>Student Group</td>
<td>CHESSN</td>
</tr>
<tr>
<td>Student Group Name</td>
<td>Student Title</td>
</tr>
<tr>
<td>Student Group Name</td>
<td>First Name</td>
</tr>
<tr>
<td>Student Group Name</td>
<td>Student Surname</td>
</tr>
<tr>
<td>Student Group Address</td>
<td>Postal Address line 1</td>
</tr>
<tr>
<td>Data Grouping</td>
<td>Data Element</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Student Details Address</td>
<td>Postal Address line 2</td>
</tr>
<tr>
<td>Student Details Address</td>
<td>Postal Address – Scholarship Postcode</td>
</tr>
<tr>
<td>Student Details Address</td>
<td>Postal Address Town/Suburb</td>
</tr>
<tr>
<td>Student Details Address</td>
<td>Postal Address – State code</td>
</tr>
<tr>
<td>Student Details Address</td>
<td>Postal Address - Country name</td>
</tr>
<tr>
<td>Scholarship Group Scholarship</td>
<td>Commonwealth Scholarship type code</td>
</tr>
<tr>
<td>Data Grouping</td>
<td>Data Element</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>Scholarship Group Scholarship</td>
<td>Commonwealth Scholarship status code</td>
</tr>
<tr>
<td>Scholarship Group Scholarship</td>
<td>Commonwealth Scholarship termination reason code</td>
</tr>
<tr>
<td>Scholarship Group Scholarship</td>
<td>Scholarship Variation code</td>
</tr>
<tr>
<td>Scholarship Group Scholarship</td>
<td>Reporting Year</td>
</tr>
<tr>
<td>Data Grouping</td>
<td>Data Element</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Scholarship Group</td>
<td>Reporting Period</td>
</tr>
</tbody>
</table>
D. Web Service Schemas

For XML elements that are concerned with the exchange of information between clients, the format and names of data elements are based on the Australian Standard 4590-1999 and ISO Standard 11179.

D.1 BaseTypes.xsd
This is a file containing all base (simple) types and elements
Figure 19: Simple Types
<?xml version="1.0" encoding="utf-8" ?>
   elementFormDefault="qualified" version="1.1">
  <xs:annotation>
    <xs:documentation>
      <status>DRAFT</status>
      <releaseDate>July 2007</releaseDate>
      <copyright>Commonwealth of Australia 2007</copyright>
      <notice>This work is subject to the laws of copyright. You may download, display, print and reproduce this material in unaltered form only (retaining this notice) for your personal, non-commercial use or use within your organisation. All other rights are reserved.
      </notice>
    </xs:documentation>
  </xs:annotation>
  <xs:simpleType name="GivenName">
    <xs:annotation>
      <xs:documentation>The first name of the person, also known as Christian name. The format is based on Clause 3.3 in AS4590-1999.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="15" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="OtherGivenName">
    <xs:annotation>
      <xs:documentation>Represents any other names a person has such as middle names. The format is based on Clause 3.3 in AS4590-1999.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="^[^\x00-\x1f\x21-\x25\x2a-\x2b\x3a-\x40\x5c-\x60\x7b-\xff]*$" />
      <xs:minLength value="1" />
      <xs:maxLength value="40" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="PersonName">
    <xs:annotation>
      <xs:documentation>Represents a person's full name.</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
<xs:pattern value="^[^x00-\x1f\x21-\x25\x2a-\x2b\x3a-\x40\x5e-\x7b-\xff]*$" />

<xs:maxLength value="100" />
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="FamilyName">
  <xs:annotation>
    <xs:documentation>Represents a persons family name or surname. If a person has only one name, it should go in this field. The format is based on Clause 3.4 in AS4590-1999.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^x00-\x1f\x21-\x25\x2a-\x2b\x3a-\x40\x5e-\x7b-\xff]*$" />
    <xs:maxLength value="30" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="AddressLine">
  <xs:annotation>
    <xs:documentation>Defines an address line used for capturing addresses.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^x00-\x1f\x21-\x25\x2a-\x2b\x3a-\x40\x5e-\x7b-\xff]*$" />
    <xs:maxLength value="38" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="CountryName">
  <xs:annotation>
    <xs:documentation>The name of a country.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^x00-\x1f\x21-\x25\x2a-\x2b\x3a-\x40\x5e-\x7b-\xff]*$" />
    <xs:maxLength value="46" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="CountryCode">
  <xs:annotation>
    <xs:documentation>A 4-digit country code. See appendix G in DESTPAC.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="\d\d\d\d" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PostCode"
<xs:simpleType name="PostalCode">
  <xs:annotation>
    <xs:documentation>Defines an address postal code based on clause 8.14 in AS4590-1999.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^\x00-\x1f|\x21-\x25|x2a-\x2b|x3a-\x40|x5c-\x60|x7b-\xff]*$"/>
    <xs:maxLength value="12" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StateCode">
  <xs:annotation>
    <xs:documentation>Defines an Australian address state or territory code based on DESTPAC element 467.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="NSW|QLD|VIC|TAS|ACT|NT|SA|WA|AAT" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ScholarshipVariationCode">
  <xs:annotation>
    <xs:documentation>A 1-digit code representing reason for revision.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[012]" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="LocalityName">
  <xs:annotation>
    <xs:documentation>The full name of the general locality containing the specific address. This will normally be the name of a town or suburb. Clause 8.11 AS-4590.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^\x00-\x1f|\x21-\x25|x2a-\x2b|x3a-\x40|x5c-\x60|x7b-\xff]*$" />
    <xs:maxLength value="46" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="StudentChessn">
  <xs:annotation>
    <xs:documentation>The Commonwealth Higher Education Student Support Number.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[1-9][1][0-9][9]*" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType>
  <xs:annotation>
    <xs:documentation>The student number allocated to a person when they attended a HEP (Higher Education Provider).</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[^\x00-\x1f\x21-\x25\x2a-\x2b\x3a-\x40\x5c-\x7b-\xff]*$" />
    <xs:minLength value="1" />
    <xs:maxLength value="10" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="LocationTypeCode">
  <xs:annotation>
    <xs:documentation>A code which identifies whether or not the student location code is in Australia or other countries.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="^[XA90]" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="LocationCode">
  <xs:annotation>
    <xs:documentation>An Australian postcode or country code.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="d(4)" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SubmissionNumber">
  <xs:annotation>
    <xs:documentation>Submission Number</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="RevisionNumber">
  <xs:annotation>
    <xs:documentation>Revision Number</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:int">
    <xs:minInclusive value="1" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:pattern value="d\d\d\d" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="RecordId">
  <xs:restriction base="xs:string">
    <xs:minLength value="1" />
    <xs:maxLength value="20" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="RequestStatusCode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Success" />
    <xs:enumeration value="Duplicate" />
    <xs:enumeration value="Failure" />
    <xs:enumeration value="Process" />
    <xs:enumeration value="Archive" />
    <xs:enumeration value="Submitted" />
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="TransactionStatusCode">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Failure" />
    <xs:enumeration value="Warning" />
  </xs:restriction>
</xs:simpleType>
<xs:restriction base="xs:string">
  <xs:enumeration value="Success"/>
  <xs:enumeration value="Ignored"/>
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="FilterTransactionStatusCode">
  <xs:annotation>
    <xs:documentation></xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="All"/>
    <xs:enumeration value="Success"/>
    <xs:enumeration value="Failure"/>
    <xs:enumeration value="Warning"/>
    <xs:enumeration value="Ignored"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="MessageCode">
  <xs:annotation>
    <xs:documentation>The lookup code of a message.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:int"/>
</xs:simpleType>

<xs:simpleType name="MessageDescription">
  <xs:annotation>
    <xs:documentation>Text description of the message.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:maxLength value="255"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="MessageSeverity">
  <xs:annotation>
    <xs:documentation>Indicates the severity type of the message.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="Error"/>
    <xs:enumeration value="Warning"/>
    <xs:enumeration value="Information"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="SchemaVersion">
<xs:annotation>
  <xs:documentation>Schema version.</xs:documentation>
</xs:annotation>

<xs:restriction base="xs:string">
  <xs:maxLength value="10" />
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="ScholarshipRecordStatus">
  <xs:annotation>
    <xs:documentation>A code indicating status of a scholarship record.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="All" />
    <xs:enumeration value="Valid" />
    <xs:enumeration value="Invalid" />
    <xs:enumeration value="Deleted" />
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ReturnToSenderFlag">
  <xs:annotation>
    <xs:documentation>A code indicating status of a ReturnToSender.</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:enumeration value="All" />
    <xs:enumeration value="Yes" />
    <xs:enumeration value="No" />
  </xs:restriction>
</xs:simpleType>
</xs:schema>
D.2 ComplexTypes.xsd

<x:schema verify="true" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="LoadScholarshipSubmissionIn" id="LoadScholarshipSubmissionIn">
    <xs:annotation>
      <xs:documentation>Contains details about the scholarship the student is receiving.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="RecordId" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="StudentNumber" type="StudentHepNumber" minOccurs="1" maxOccurs="1" />
      <xs:element name="Chessn" type="StudentChessn" minOccurs="1" maxOccurs="1" />
      <xs:element name="Title" type="PersonTitle" minOccurs="1" maxOccurs="1" />
      <xs:element name="GivenName" type="GivenName" minOccurs="1" maxOccurs="1" />
      <xs:element name="FamilyName" type="FamilyName" minOccurs="1" maxOccurs="1" />
      <xs:element name="PostalAddress" type="Address" minOccurs="0" maxOccurs="1" />
      <xs:element name="StudentScholarship" type="StudentScholarshipDetails" />
    </xs:sequence>
  </xs:complexType>

  <xs:element name="LoadSubmissionOut">
    <xs:annotation>
      <xs:documentation>Output transaction data returned for all load methods.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="RecordId" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionStatus" type="TransactionStatus" minOccurs="1" maxOccurs="1" />
      <xs:element name="InputStream" type="xs:base64Binary" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>

  <xs:element name="SubmissionStatusOut">
    <xs:annotation>
      <xs:documentation>A structure containing transaction status for a submission</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="TotalTransactions" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionsCompleted" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionsInError" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionsInWarning" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionsExecuting" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionsInSubmit" type="xs:int" minOccurs="1" maxOccurs="1" />
      <xs:element name="TransactionsIgnored" type="xs:int" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
<xs:complexType name="SearchScholarshipSubmissionIn">
    <xs:annotation>
        <xs:documentation>Input transaction data required in the SearchScholarships method.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="SubmissionYearFrom" type="xs:gYear" minOccurs="0" maxOccurs="1" />
        <xs:element name="SubmissionYearTo" type="xs:gYear" minOccurs="0" maxOccurs="1" />
        <xs:element name="ReportingPeriod" type="ReportingPeriod" minOccurs="0" maxOccurs="1" />
        <xs:element name="ClientOrganisationCode" type="ClientOrganisationCode" minOccurs="0" maxOccurs="1" />
        <xs:element name="ClientRequestId" type="xs:string" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="SearchSubmissionOut">
    <xs:annotation>
        <xs:documentation>Output transaction data returned in the SearchSubmission method.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="RequestId" type="RequestId" minOccurs="1" maxOccurs="1" />
        <xs:element name="ClientOrganisationCode" type="ClientOrganisationCode" minOccurs="1" maxOccurs="1" />
        <xs:element name="LoadType" type="xs:string" minOccurs="1" maxOccurs="1" />
        <xs:element name="LoadStatus" type="xs:string" minOccurs="1" maxOccurs="1" />
        <xs:element name="SubmissionYear" type="xs:gYear" minOccurs="1" maxOccurs="1" />
        <xs:element name="LoadCount" type="xs:int" minOccurs="1" maxOccurs="1" />
        <xs:element name="SubmissionRevisionNumber" type="xs:int" minOccurs="1" maxOccurs="1" />
        <xs:element name="CreateLogonName" type="xs:string" minOccurs="1" maxOccurs="1" />
        <xs:element name="CreateDateTime" type="xs:dateTime" minOccurs="1" maxOccurs="1" />
        <xs:element name="UpdateLogonName" type="xs:string" minOccurs="1" maxOccurs="1" />
        <xs:element name="UpdateDateTime" type="xs:dateTime" minOccurs="1" maxOccurs="1" />
    </xs:sequence>
</xs:complexType>

<xs:complexType name="SearchScholarshiIn">
<xs:element name="Chessn" type="StudentChessn" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipCode" type="ClsScholarshipCode" minOccurs="0" maxOccurs="1" />
<xs:element name="ReportingYear" type="ReportingYear" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipStatusCode" type="ClsScholarshipStatusCode" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipTerminationReasonCode" type="ClsScholarshipTerminationReasonCode" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipDeleteFlag" type="xs:boolean" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipInvalidFlag" type="xs:boolean" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipCreateDateTime" type="xs:date" minOccurs="0" maxOccurs="1" />
<xs:element name="ScholarshipUpdateDateTime" type="xs:date" minOccurs="0" maxOccurs="1" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="SearchScholarshipOut">
  <xs:annotation>
    <xs:documentation>Output transaction data returned in SearchScholarships method.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="StudentId" type="xs:int" minOccurs="0" maxOccurs="1" />
    <xs:element name="StudentNumber" type="StudentHepNumber" minOccurs="1" maxOccurs="1" />
    <xs:element name="Chessn" type="StudentChessn" minOccurs="0" maxOccurs="1" />
    <xs:element name="Title" type="PersonTitle" minOccurs="0" maxOccurs="1" />
    <xs:element name="GivenName" type="GivenName" minOccurs="0" maxOccurs="1" />
    <xs:element name="FamilyName" type="FamilyName" minOccurs="0" maxOccurs="1" />
    <xs:element name="PostalAddress" type="Address" minOccurs="0" maxOccurs="1" />
    <xs:element name="ReportingPeriod" type="ReportingPeriod" minOccurs="0" maxOccurs="1" />
    <xs:element name="ReportingYear" type="ReportingYear" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipCode" type="ClsScholarshipCode" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipStatusCode" type="ClsScholarshipStatusCode" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipTerminationReasonCode" type="ClsScholarshipTerminationReasonCode" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipDeleteFlag" type="xs:boolean" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipInvalidFlag" type="xs:boolean" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipCreateDateTime" type="xs:date" minOccurs="0" maxOccurs="1" />
    <xs:element name="ScholarshipUpdateDateTime" type="xs:date" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RequestControlTable">
  <xs:annotation>
    <xs:documentation>Contains control data for all types of requests.</xs:documentation>
  </xs:annotation>
</xs:complexType>
<xs:sequence>
  <xs:element name="RequestId" type="RequestId" minOccurs="1" maxOccurs="1" />
  <xs:element name="ClientOrganisationCode" type="ClientOrganisationCode" minOccurs="1" maxOccurs="1" />
  <xs:element name="RequestLocalDateTime" type="xs:dateTime" minOccurs="1" maxOccurs="1" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="ResponseControlTable">
  <xs:annotation>
    <xs:documentation>Contains control data for all responses from the server.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="RequestId" type="RequestId" minOccurs="1" maxOccurs="1" />
    <xs:element name="ClientOrganisationCode" type="ClientOrganisationCode" minOccurs="1" maxOccurs="1" />
    <xs:element name="RequestStatus" type="RequestStatus" minOccurs="1" maxOccurs="1" />
    <xs:element name="CurrentDateTime" id="CurrentDateTime" type="xs:date" minOccurs="1" maxOccurs="1" />
    <xs:element name="ReceivedDateTime" id="ReceivedDateTime" type="xs:dateTime" minOccurs="1" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="RequestStatus">
  <xs:annotation>
    <xs:documentation>Contains the status of the request and any messages (informational, error or otherwise) associated with the request.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="StatusCode" type="RequestStatusCode" minOccurs="1" maxOccurs="1" />
    <xs:element name="Messages" type="ArrayOfMessage" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="TransactionStatus">
  <xs:annotation>
    <xs:documentation>Contains the status of the transactions.</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="StatusCode" type="TransactionStatusCode" minOccurs="1" maxOccurs="1" />
    <xs:element name="Messages" type="ArrayOfMessage" minOccurs="0" maxOccurs="1" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="ArrayOfMessage">
  <xs:annotation>
    <xs:documentation>Container for all messages.</xs:documentation>
  </xs:annotation>
</xs:complexType>
<xs:annotation>
    <xs:documentation>An individual message.</xs:documentation>
</xs:annotation>
<xs:sequence>
    <xs:element name="Code" type="MessageCode" minOccurs="1" maxOccurs="1" />
    <xs:element name="Description" type="MessageDescription" minOccurs="1" maxOccurs="1" />
    <xs:element name="Severity" type="MessageSeverity" minOccurs="1" maxOccurs="1" />
    <xs:element name="BusinessKeys" type="ArrayOfElements" minOccurs="0" maxOccurs="1" />
    <xs:element name="ErrorElements" type="ArrayOfElements" minOccurs="0" maxOccurs="1" />
    <xs:element name="Validation" type="ValidationType" minOccurs="0" maxOccurs="1" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="Address">
    <xs:annotation>
        <xs:documentation>A structure for generic addresses based on the AS4590 standard. The standard splits up addresses into 13 clauses, but suggests they can be combined into several address lines. Please use the LocalityName, StateCode, PostCode and CountryName elements if possible and avoid combining these fields into the AddressLine2 element.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="AddressLine1" type="AddressLine" minOccurs="0" maxOccurs="1" />
        <xs:element name="AddressLine2" type="AddressLine" minOccurs="0" maxOccurs="1" />
        <xs:element name="LocalityName" type="LocalityName" minOccurs="0" maxOccurs="1" />
        <xs:element name="StateCode" type="StateCode" minOccurs="0" maxOccurs="1" />
        <xs:element name="PostCode" type="PostCode" minOccurs="0" maxOccurs="1" />
        <xs:element name="CountryCode" type="CountryCode" minOccurs="0" maxOccurs="1" />
        <xs:element name="CountryName" type="CountryName" minOccurs="0" maxOccurs="1" />
    </xs:sequence>
</xs:complexType>
D.3 ScholarshipWebServiceRequestAndResponse.xsd

D.3.1.1 LoadScholarshipSubmission.xsd

```xml
<?xml version="1.0" encoding="utf-8" ?>
  elementFormDefault="qualified" version="1.1">
  <xs:include schemaLocation="../DataElements/TechnicalComplexTypes.xsd" />
  <xs:include schemaLocation="../DataElements/BusinessComplexTypes.xsd" />

  <xs:element name="LoadScholarshipSubmissionInRequest">
    <xs:annotation>
      <xs:documentation>A request to Load Scholarship.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1" />
        <xs:element name="TransactionData" type="ArrayOfLoadScholarshipSubmissionIn" />
      </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="SubmissionResults" type="ArrayOfLoadSubmissionOut" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```
D.3.1.2 GetScholarshipTransactionResultRequest.xsd

```xml
<?xml version="1.0" encoding="utf-8" ?>
    elementFormDefault="qualified" version="1.1">
    <xs:include schemaLocation="../DataElements/TechnicalComplexTypes.xsd" />  
    <xs:include schemaLocation="../DataElements/BusinessComplexTypes.xsd" />

    <xs:element name="GetScholarshipTransactionResultRequest">
        <xs:annotation>
            <xs:documentation>Resturns submission results based on the Transaction Status. Returns all if TransactionStatus is not provided</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1" />
                <xs:element name="RequestIdList" type="ArrayOfRequestId" minOccurs="1" maxOccurs="1" />
                <xs:element name="TransactionStatus" type="FilterTransactionStatusCode" minOccurs="0" 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="RequestIdList" type="ArrayOfRequestId" minOccurs="1" maxOccurs="1" />
                <xs:element name="TransactionStatus" type="FilterTransactionStatusCode" minOccurs="0" maxOccurs="1" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```
D.3.1.3  GetSubmissionStatusRequest.xsd

<?xml version="1.0" encoding="utf-8" ?>
    elementFormDefault="qualified" version="1.1">
  <xs:include schemaLocation="../DataElements/TechnicalComplexTypes.xsd" />
  <xs:include schemaLocation="../DataElements/BusinessComplexTypes.xsd" />
  <xs:element name="GetSubmissionStatusRequest">
    <xs:annotation>
      <xs:documentation></xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1" />
        <xs:element name="SubmissionRequestId" type="RequestId" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="GetSubmissionStatusResponse">
    <xs:annotation>
      <xs:documentation>Contains details about transaction status for a submission</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ResponseControlData" type="ResponseControlTable" minOccurs="1" maxOccurs="1" />
        <xs:element name="GetSubmissionStatusOut" type="SubmissionStatusOut" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
D.3.1.4 SearchScholarshipDataRequest.xsd

```xml
<?xml version="1.0" encoding="utf-8" ?>
  elementFormDefault="qualified" version="1.1">
  <xs:include schemaLocation="../DataElements/TechnicalComplexTypes.xsd" />
  <xs:include schemaLocation="../DataElements/BusinessComplexTypes.xsd" />
  <xs:include schemaLocation="../Structures/ScholarshipSubmission/StudentEntities.xsd" />

  <xs:element name="SearchScholarshipDataRequest">
    <xs:annotation>
      <xs:documentation>Request to retrieve submission details</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1" />
        <xs:element name="SearchScholarshipIn" type="SearchScholarshipIn" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="SearchScholarshipDataResponse">
    <xs:annotation>
      <xs:documentation>Response containing submission details</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ResponseControlData" type="ResponseControlTable" minOccurs="1" maxOccurs="1" />
        <xs:element name="SearchScholarshipOut" type="ArrayOfSearchScholarshipOut" minOccurs="0" />
        <xs:element name="TotalCount" type="xs:int" minOccurs="0" maxOccurs="1" />
        <xs:element name="TotalPageCount" type="xs:int" minOccurs="0" maxOccurs="1" />
        <xs:element name="PageSize" type="xs:int" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
D.3.1.5  SearchScholarshipSubmissionRequest.xsd

<?xml version="1.0" encoding="utf-8" ?>

elementFormDefault="qualified" version="1.1">
<xs:include schemaLocation="../DataElements/TechnicalComplexTypes.xsd" />
<xs:include schemaLocation="../DataElements/BusinessComplexTypes.xsd" />

<xs:element name="SearchScholarshipSubmissionRequest">
  <xs:annotation>
    <xs:documentation>Request to Retrieve scholarship submission details</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="RequestControlData" type="RequestControlTable" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SearchScholarshipSubmissionIn" type="SearchScholarshipSubmissionIn" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="SearchScholarshipSubmissionsResponse">
  <xs:annotation>
    <xs:documentation>Response containing scholarship submissions details</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ResponseControlData" type="ResponseControlTable" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SearchScholarshipSubmissionOut" type="ArrayOfSearchSubmissionOut" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>