

Preparing Data Files for Custom Imports and Feeds into Elements

Prepared by:

Dave Budenberg

dave@symplectic.co.uk

10 March 2015

Symplectic Limited, 4 Crinan Street, London,
N1 9XW, United Kingdom.

Contents

1 Summary

This document gives guidance and advice about the preparation and format of data files that are to be used for custom imports and feeds into Elements.

(See an [overview of the process of migrating data into Elements](#) for a higher level summary and details of the Standard Element Importer.)

1.1 Background

When Elements is first implemented data is often populated from existing systems within the organisation. These may take the form of one-off imports, or may involve repeated feeds of data from external systems.

Data can be imported using a variety of approaches, these include:

1. Using the Standard Elements Importer tool
2. Using custom import applications developed by Symplectic
3. Using custom applications developed by customers, [using the Elements API](#) (with a [sample application](#) for reference).

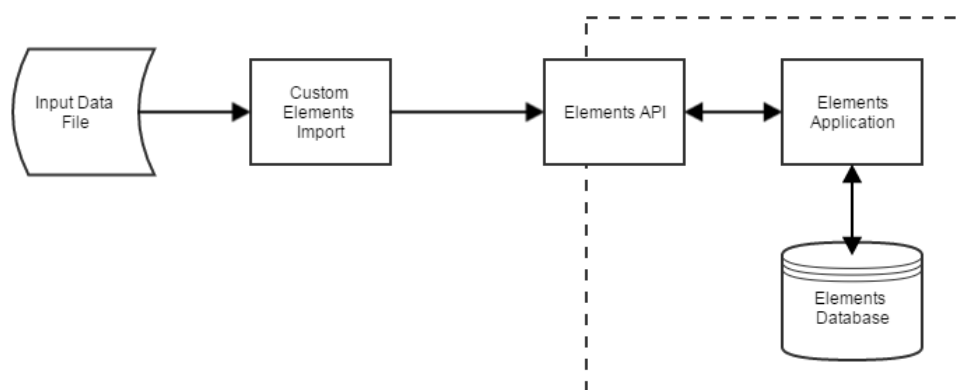
All these approaches make use of the Elements API, ensuring that they are not tied to individual versions of Elements.

This document will focus on data files suitable for use with custom import/feed applications that are developed by Symplectic (2 above), but may also be helpful for organisations preparing their own import/feed applications (3 above).

2 Introduction

2.1 Typical Application Structure

A typical custom import application follows the structure below:



The custom Elements import application performs the following processing:

- Reads the input data file
- Identifies the destination object type to be created/updated (e.g. grant object, course taught object etc)
- Maps the source data values to the properties of the chosen object
- Passes the data to Elements using the Elements API.

For some complex imports it may be necessary to use more than one input file, but this can (and should) be avoided in most cases.

2.2 Input Data File Formats

The input data file will typically be provided in one of the following formats:

- comma separated values (csv)
- Extensible markup language (xml).

The format chosen will depend on:

- Ease of preparation
Usually, choose the format that your organisation is more familiar with
- The complexity of the data structure to be imported
Complex data structures are usually easier to prepare using xml rather than csv (it may be impossible to prepare a single file when using a csv format).

In order to ensure that all international characters are supported input files should all use UTF-8 character encoding.

3 Customer Organisation Steps

Prior to the development of a custom import the customer organisation should:

1. Identify what data is to be imported
2. Identify where data is to be updated
3. Identify the Elements data source to be used
4. Identify how often data is to be imported
5. Identify the destination category
6. Prepare Elements object type definitions
7. Define data mapping
8. Identify a unique proprietary id value for each object to be imported
9. Identify how links will be created
10. Prepare the input data file for import (using either csv or xml)
11. Prepare an Elements environment to be used for testing the import, with Elements API operational (if data is to be imported into user profiles the API endpoint will need a SSL certificate)
12. Test the custom import application

3.1 Identify what data is to be imported

Consider:

Why is the data required

Data should only be loaded into Elements if there is a reason to do so. Reasons may include:

- Online access to the data is required
- Data is required for reporting and analysis purposes
- Data is to be fed to other applications/systems (e.g. Repository, VIVO, Profiles etc).

Where the data currently resides

Data may exist within one or more existing applications. If data is not available it cannot be imported.

How data will be extracted

Extracting data may require additional analysis and development effort.

Data quality

Data to be imported should be:

- Accurate and consistent
- Complete

- Deduplicated

If more than one source exists consideration must be given to ensuring that inconsistent, ambiguous or duplicate data is resolved.

3.2 Identify where data is to be updated

It is important to be clear about where data that has been imported into Elements is updated - problems can occur if data can be updated both external to Elements and within Elements. Also, if data is updated outside Elements, how are those updates to be made available in Elements (if this is needed)?

By choosing an appropriate data source you can ensure that no user is able to edit the imported data. The only data that can be edited by users in Elements has a data source of 'Manual' - import using any other data source and users will not be able to edit the data.

If data is being imported from an 'authoritative' upstream system that data should not usually be maintained in Elements.

[Note: From v4.14 object types will include support for field-level locking - this will provide the ability to stop individual fields from being edited for records with a 'Manual' data source. This will provide mechanisms for more complex data feed scenarios to be supported.]

3.3 Identify the Elements data source to be used

All data loaded into Elements has an associated data source. Available data sources are:

- Institutional Source 1
- Institutional Source 2
- Manual.

If imported data is to be edited by users the 'Manual' data source must be used.

3.4 Identify how often data is to be imported

In some cases data will be imported into Elements just once when setting up Elements. In this situation we do not need to consider some of the complicating factors that affect ongoing feeds or regular imports.

Ongoing feeds or regular imports require additional considerations, primarily to consider how to handle the same data being imported more than once. Additional logic is needed to extract subsets of data if a complete re-import is not being performed.

A number of scenarios should be considered:

Complete re-import each time

If a source system continues to have data updated, and this needs to be reflected in

Elements, the simplest way to achieve this may be to import all the data every time an import is run (perhaps every week). The objective here is to replace any existing objects and add new objects where required.

In order to achieve this we need an identifier for each record which enables us to match records reliably. Elements supports this through the use of an internal proprietary-id value which uniquely identifies a record of a given category and data source. When importing data, if a record with the same category, data source and proprietary-id exists it will be replaced by the incoming record, otherwise a new record will be created.

This works fine for records that are not updated within Elements, but may not work as desired if data is being updated in Elements (when an import is run an updated record will be replaced with a matched external one, losing the update(s) made within Elements).

Import new records only

Importing only new records from the source system avoids problems of overwriting of existing data, but does not provide a mechanism for changes in the source system to be propagated to Elements.

Import changed records only

Importing only changed records can work well if no changes are allowed in Elements.

3.5 Identify the destination category

Usually it is clear which destination category (Publication, Grant etc) should be used for a data import. However, there may be times when it is not so obvious - don't forget about user profiles (User category).

3.6 Prepare Elements object type definitions

Elements provides a standard set of object type definitions within each category. For most imports these will provide most of the data fields to be populated. However each organisation may want to adjust existing object types by adding or removing fields from the definition to exactly match their requirements. In some cases completely new object types (custom object types) may be required.

3.7 Define data mapping

This involves identifying how each Elements data value will be populated from the source data. At the simplest level this may just be a mapping of a text value from the source data to a text field in the destination object type. Some fields may be more complex - a address-list field may involve taking data from a number of source data fields.

An Excel spreadsheet should be completed for the import, ensuring that there is an entry for each field in the destination object type and each source value.

3.8 Identify a proprietary-id value for each object to be imported

Elements uses an internal proprietary-id value to uniquely identify a record of a given category and data source. This is a key that is used to match import records to records which may already exist within Elements, determining whether a new record is created or an existing one is overwritten.

Whenever a record is being imported it should have a proprietary-id specified, typically built from one or more source data values (e.g. a from a primary key value in the source system). For ongoing feeds this is an absolute requirement - a one off feed can generate a proprietary-id if none is supplied.

3.9 Identify how links will be created

Elements is a user-centric application, and imported data will usually need to be linked with an Elements user. This link information is usually specified by providing the proprietary-id or username for an Elements user in the import data. These data values are typically maintained in Elements using the HR Feed.

If links are required to other objects the proprietary-id of the other objects will be required (together with the type of link).

3.10 Prepare the input data file for import

At this stage all the data values required for the import have been identified, together with the source of the data.

An input data file is created by extracting the required data values and generating a file of the chosen format (csv or xml). and formatting records using formats similar to those shown in the examples in the [Appendix](#).

Each record should contain:

- values for each source field
- values to specify each link.

3.11 Prepare an Elements environment

In order to test the import a test environment should be provided. This may be an independent implementation of Elements which uses a copy of the production Elements database.

Note that the Elements API must be operational (if data is to be imported into user profiles the API endpoint will need a SSL certificate).

It is often helpful to provide a backup of the Elements database to Symplectic to enable initial testing to be performed by them.

3.12 Test the custom import application

The supplied application will be tested to verify that it operates as required. Amendments will be discussed with Symplectic, who will provide updated software to address items.

4 Symplectic Steps

Symplectic will provide the following:

1. Assist and advise the customer organisation
2. Prepare and test a custom import application
3. Provide application for user testing
4. Amend the application based on feedback from user testing

4.1 Assist and advise the customer organisation

Symplectic will advise and assist the customer organisation in following the steps shown above.

4.2 Prepare and test a custom import application

Based on the following deliverables, Symplectic will prepare and test a custom import application:

1. Completed import mapping definition spreadsheet
2. Input data file

4.3 Provide application for user testing

A complete and tested application will be provided, including simple written instructions. The instructions will cover:

- specification of customisable parameters
- all outputs, including log files
- how to invoke an import

4.4 Amend the application based on feedback from user testing

Based on discussions with the customer organisation following user testing amendments will be made to the application to resolve issues raised.

5 User Profile Imports

User profile imports use the same principles as other data imports, but a number of other factors need to be considered.

5.1 Special Considerations

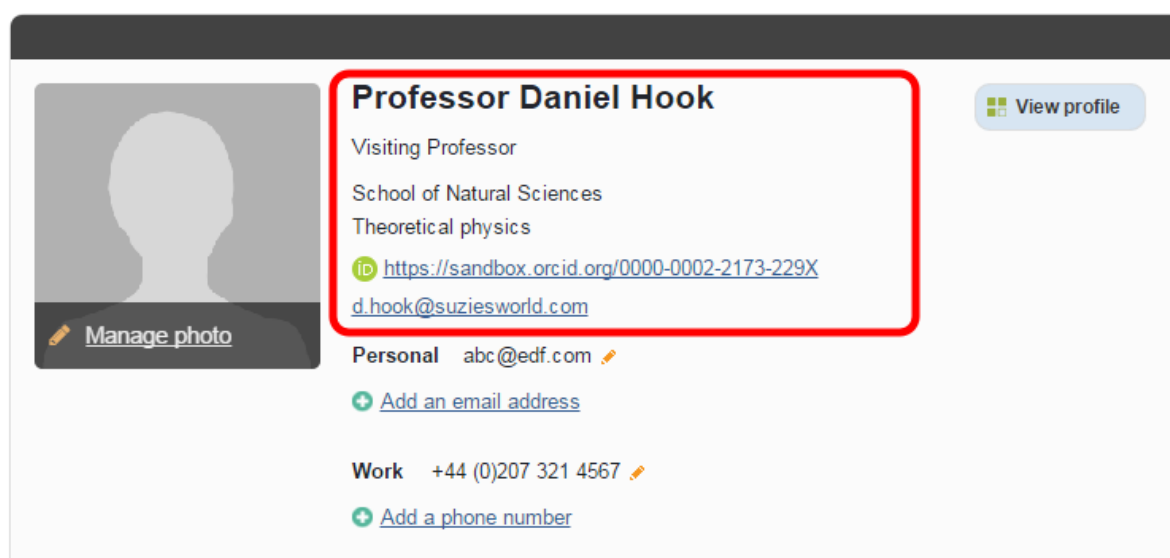
Users cannot be created using a user profile import (an Elements HR feed should be used to create users).

All users have a user profile record so a user profile import will always overwrite data.

Unlike imports for other categories a user profile import will only update fields for which values are specified in the input file. The data types of user profile fields are shown in [Appendix 2](#). These data types are more complex than most other data types, and it is recommended that import files for user profiles use xml formats.

Data that is not updatable

Some of the data shown on the User Profile page exists as part of a User, not a user profile. These data values (shown in the red box below) cannot be updated using a custom import (use the HR Feed for this).



Data Source

All User Profile imports use the 'Manual' data source.

proprietary-id

The proprietary id value for a user profile import is the user proprietary id value.

Links

User profile data is associated with a single user. No link information is required.

6 Appendix - Elements Data Types

Elements uses a consistent data model for all objects within elements. All object types are defined using this model, based on the list of field types below. Details of internal elements field types used by User Profiles can be found [below](#).

6.1 Standard Elements Data Types

The following data types are used for all objects, and can also be used when defining custom object types. For each data type examples show how a value of that type should be specified in an input data file, using csv and xml formats.

text data type

<i>Description</i>	This contains a single text string
<i>Example uses</i>	Titles, descriptions, names etc
<i>csv example</i>	, "3D Jigsaws - How biomolecules fit together",
<i>xml example</i>	<title>3D Jigsaws - How biomolecules fit together</title>

date data type

<i>Description</i>	This contains a date value
<i>Example uses</i>	Start date, End date
<i>csv example</i>	, "2014-07-23",
<i>xml example</i>	<start-date>2014-07-23</start-date>
<i>or</i>	<start-date><year>2014</year><month>07</month></start-date>

choice data type

<i>Description</i>	A text string, that must be one of a choice of values configured in Elements
<i>Example uses</i>	Status values (Submitted, Accepted, Published)
<i>csv example</i>	, "Published",
<i>xml example</i>	<status>Published</status>

integer data type

<i>Description</i>	An integer value
<i>Example uses</i>	Number of students enrolled on a course
<i>csv example</i>	, "27",
<i>xml example</i>	<number-of-students>27</number-of-students>

boolean data type

<i>Description</i>	An boolean (true/false) value
<i>Example uses</i>	Whether a user was invited to present at a conference
<i>csv example</i>	, "true",
<i>xml example</i>	<invited>true</invited>

number data type

<i>Description</i>	A decimal value
<i>Example uses</i>	Number of credits for a course
<i>csv example</i>	, "4.5",
<i>xml example</i>	<number-of-credits>4.5</number-of-credits>

list data type

<i>Description</i>	A list of text values
<i>Example uses</i>	
<i>csv example</i>	, "blue,green,yellow",
<i>xml example</i>	<available-colours> <colour>blue</colour> <colour>green</colour> <colour>yellow</colour> </available-colours>
<i>or</i>	<available-colours>blue,green,yellow</available-colours>

isbn-10 data type

<i>Description</i>	A valid ISBN-10 value
<i>Example uses</i>	ISBN-10
<i>csv example</i>	, "0-19-852663-6",

xml example <isbn-10>0-19-852663-6</isbn-10>

isbn-13 data type

Description A valid ISBN-13 value

Example uses ISBN-13

csv example ,”978-1-86197-876-9”,

xml example <isbn-13>978-1-86197-876-9</isbn-13>

issn data type

Description A valid ISSN value

Example uses ISSN

csv example ,”0028-0836”,

xml example <issn>0028-0836</issn>

person-list data type

Description A list of people’s names

Example uses Co-contributors to a course taught

csv example ,”Jones,HS;Smith,JK”,

xml example <co-contributors>
 <person><lastname>Jones</lastname><initials>HS</initials></person>
 <person><lastname>Smith</lastname><initials>JK</initials></person>
 </co-contributors>

person data type

Description A person’s name

Example uses Student name, supervisor name

csv example ,”Ferguson,SD”,

xml example <student><lastname>Ferguson</lastname><initials>SD</initials></student>

url data type

<i>Description</i>	A person's name
<i>Example uses</i>	Student name, supervisor name
<i>csv example</i>	, "www.symplectic.co.uk",
<i>xml example</i>	<url>www.symplectic.co.uk</url>

keyword-list data type

<i>Description</i>	A list of text values
<i>Example uses</i>	
<i>csv example</i>	, "Red,Green,Blue",
<i>xml example</i>	<pre><available-colours> <colour>blue</colour> <colour>green</colour> <colour>yellow</colour> </available-colours></pre>

doi data type

<i>Description</i>	A valid DOI value
<i>Example uses</i>	DOI
<i>csv example</i>	, "dx.doi.org/10.1103/PhysRevLett.95.086601",
<i>xml example</i>	<doi>dx.doi.org/10.1103/PhysRevLett.95.086601</doi>

pagination data type

<i>Description</i>	A valid Pagination value
<i>Example uses</i>	Pagination of an article in a journal
<i>csv example</i>	, "234-241",
<i>xml example</i>	<pagination>234-241</pagination>
<i>or</i>	<pagination><from>234</from><to>241</to></pagination>

address-list data type

<i>Description</i>	A list of addresses (note there is no address field type)
<i>Example uses</i>	Anywhere where an address is required
<i>csv example</i>	, "",

xml example

```

<addresses>
  <address>
    <name>John Smith</name>
    <organisation>Symplectic Ltd</organisation>
    <suborganisation>PR Dept</suborganisation>
    <street>4 Crinan Street</street>
    <city>London</city>
    <state><state>
    <zip-code>N1 9XW</zip-code>
    <country>United Kingdom</country>
  </address>
  <address type="mailing">
    <name>Sherlock Holmes</name>
    <organisation>A Detective Agency</organisation>
    <suborganisation>Accounts Dept</suborganisation>
    <street>223b Baker St</street>
    <city>London</city>
    <state><state>
    <zip-code><zip-code>
    <country>United Kingdom</country>
  </address>
</addresses>

```

money data type

<i>Description</i>	A financial value, comprising an amount and a currency (one of USD, GBP, EUR, AUD, NZD, JPY, CAD). If a currency value is not specified, the Elements default will be used
<i>Example uses</i>	Value of a grant
<i>csv example</i>	, "1234;USD",
<i>xml example</i>	<pre> <money>1234</money> or <money> <amount>1234</amount> <currency>USD</currency> </money> </pre>

6.2 Appendix 2 - Internal Elements Data Types

A number of additional internal data types are used within Elements. These cannot be used with custom object types, but are used by the User Profile.

If data is to be imported into User Profiles, the following additional data types are used. For each data type examples show how a value of that type should be specified in an input data file, using csv and xml formats.

academic-appointment-list data type

Description A list of academic appointments

Example uses List of academic appointments within a user profile

UI - view

Academic appointments:

Honorary Adjunct Assistant Professor, Washington University at St Louis, Jan 2010 - present
Academic Visitor, Imperial College London, Jan 2007 - present
PhD Student, Imperial College London, 2000 - 2007

UI - edit

Academic appointments:

Position *	<input type="text" value="Honorary Adjunct Assistant Professor"/>
Institution name *	<input type="text" value="Washington University at St Louis"/>
Department	<input type="text"/>
Street	<input type="text"/>
City	<input type="text"/>
State or province	<input type="text"/>
Zip code or postcode	<input type="text"/>
Country *	<input type="text" value="United States"/>
Start date *	<input type="text" value="01 Jan 2010"/>
End date	<input type="text"/>

csv example

not recommended

xml example

```
<academic-appointments>
  <academic-appointment>
    <position></position>
    <institution>
      <name></name>
      <department></department>
      <street></street>
      <city></city>
      <state></state>
      <zipcode></zipcode>
      <country></country>
    </institution>
    <start-date></start-date>
    <end-date></end-date>
  </academic-appointment>
  <academic-appointment>
    ...
  </academic-appointment>
  <academic-appointment>
    ...
  </academic-appointment>
</academic-appointments>
```

non-academic-employment-list data type

Description A list of non-academic employments

Example uses List of non-academic employments within a user profile

UI - view

Non-academic employment:

Director, Research Metrics, Digital Science, 2013 - present
Director, Symplectic Ltd, 2003 - present

UI - edit

Non-academic employment:

Position *	<input type="text" value="Director, Research Metrics"/>
Employer *	<input type="text" value="Digital Science"/>
Department	<input type="text"/>
Street	<input type="text"/>
City	<input type="text"/>
State or province	<input type="text"/>
Zip code or postcode	<input type="text"/>
Country	<input type="text"/>
Start date *	<input type="text" value="2013"/>
End date	<input type="text"/>

csv example

not recommended

xml example

```
<non-academic-employments>
  <non-academic-employment>
    <position></position>
    <employer>
      <name></name>
      <department></department>
      <street></street>
      <city></city>
      <state></state>
      <zipcode></zipcode>
      <country></country>
    </employer>
    <start-date></start-date>
    <end-date></end-date>
  </non-academic-employment>
  <non-academic-employment>
    ...
  </non-academic-employment>
</non-academic-employments>
```

degree-list data type

Description A list of degrees

Example uses List of degrees within a user profile

UI - view

Degrees:
 B.Sc Biochemistry, University of Melbourne, Australia, - Jan 1970
 M.Sc Biochemistry, University of Melbourne, Australia, - Jan 1972
 Ph.D Molecular Biology, University of Cambridge, United Kingdom, - Jan 1975
 Postdoc. Molecular and Cellular Biology, Yale University, United States, Jan 1975 - Jan 1977

UI - edit

Degrees:

Qualification *	<input type="text" value="B.Sc"/>	▼
Institution name *	<input type="text" value="University of Melbourne"/>	▲
Department	<input type="text"/>	
Street	<input type="text"/>	
City	<input type="text"/>	
State or province	<input type="text"/>	
Zip code or postcode	<input type="text"/>	
Country *	<input type="text" value="Australia"/>	
Start date	<input type="text"/>	📅
End date	<input type="text" value="01 Jan 1970"/>	📅

🗑️ ↺ ✅ ❌

csv example not recommended

xml example

```
<degrees>
  <degree>
    <qualification></qualification>
    <major-field-of-study></major-field-of-study>
    <thesis-title></thesis-title>
    <supervisor>
      <lastname></lastname>
      <initials></initials>
    </supervisor>
  </degree>
  <degree>
    ...
  </degree>
</degrees>
```

certification-list data type

Description A list of certifications




Example uses List of certifications within a user profile





UI - view

Certifications:
Microsoft, Microsoft Certified Professional (MCP), 2011 - present

UI - edit

Certifications:

Institution name *	<input type="text" value="Microsoft"/>	
City	<input type="text"/>	
Country	<input type="text"/>	
Name or title *	<input type="text" value="Microsoft Certified Professional (MCP)"/>	
Effective date *	<input type="text" value="2011"/>	
Expiry date	<input type="text"/>	
Description	<div style="border: 1px solid #ccc; height: 80px;"></div>	

csv example

not recommended

xml example

```
<certifications>
  <certification>
    <name></name>
    <effective-date></effective-date>
    <expiry-date></expiry-date>
    <description></description>
    <certifying-body>
      <name></name>
      <department></department>
      <street></street>
      <city></city>
      <state></state>
      <zipcode></zipcode>
      <country></country>
    </certifying-body>
  </certification>
  <certification>
    ...
  </certification>
</certifications>
```

postgraduate-training-list data type

Description A list of postgraduate training



Example uses List of postgraduate training within a user profile

UI - view

UI - edit

Postgraduate training:

You haven't listed any postgraduate training.

Title *	<input type="text"/>
Institution name *	<input type="text"/>
Department	<input type="text"/>
Street	<input type="text"/>
City	<input type="text"/>
State or province	<input type="text"/>
Zip code or postcode	<input type="text"/>
Country *	<input type="text"/>
Category	- ▾
Specialisation	<input type="text"/>
Supervisor last name	<input type="text"/>
Supervisor initials	<input type="text"/>
Start date	<input type="text"/> 
End date	<input type="text"/> 

✓ ✕

csv example

not recommended

xml example

```

<postgraduate-trainings>
  <postgraduate-training>
    <title></title>
    <institution>
      <name></name>
      <department></department>
      <street></street>
      <city></city>
      <state></state>
      <zipcode></zipcode>
      <country></country>
    </institution>
    <category></category>
    <specialisation></specialisation>
    <supervisor>
      <lastname></lastname>
      <initials></initials>
    </supervisor>
    <start-date></start-date>
    <end-date></end-date>
  </postgraduate-training>
  <postgraduate-training>
    ...
  </postgraduate-training>
</postgraduate-trainings>

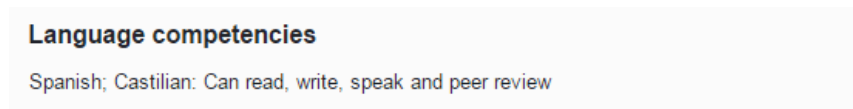
```

language-competency-list data type

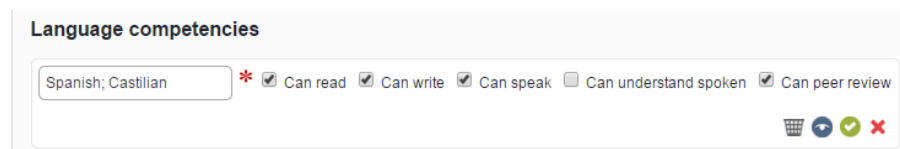
Description A list of language competencies

Example uses List of language competencies within a user profile

UI - view



UI - edit



csv example not recommended

xml example

```
<language-competencies>
  <language-competency>
    <iso-language-code></iso-language-code>
    <can-read></can-read>
    <can-write></can-write>
    <can-speak></can-speak>
    <can-understand-spoken></can-understand-spoken>
  >
  <can-peer-review></can-peer-review>
</language-competency>
<language-competency>
  ...
</language-competency>
</language-competencies>
```

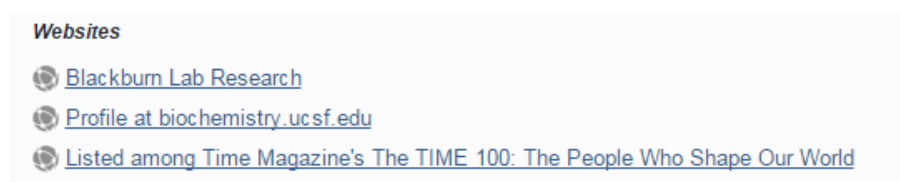
Notes Language code value should match a language code as defined in [ISO 639-2 definition](#).

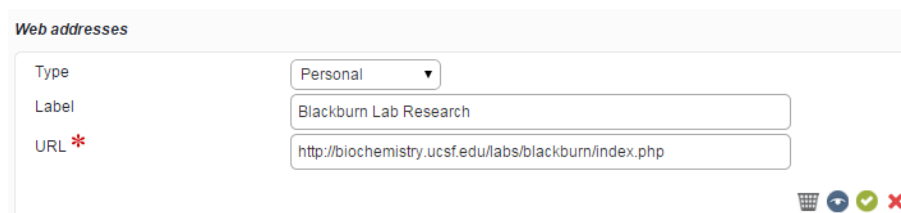
web-address-list data type

Description A list of web addresses

Example uses List of web addresses within a user profile

UI - view



UI - edit


Web addresses

Type: Personal

Label: Blackburn Lab Research

URL *: http://biochemistry.ucsf.edu/labs/blackburn/index.php

Icons: trash, undo, redo, delete

csv example

not recommended

xml example

```

<web-addresses>
  <web-address>
    <web-address-type></web-address-type>
    <label></label>
    <url></url>
  </web-address>
  <web-address>
    ...
  </web-address>
</web-addresses>

```

Notes

Web address type is optional. Supported values are:

- Personal
- Company
- Blog
- RssFeed
- Portfolio
- Twitter
- LinkedIn
- GoogleScholar
- ResearchGate
- Figshare
- Mendeley
- Other

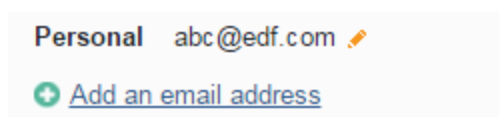
Label is optional.

email-address-list data type*Description*

A list of email addresses

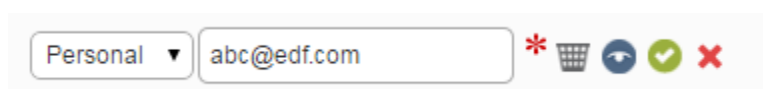
Example uses

List of email addresses within a user profile

UI - view


Personal abc@edf.com ✎

+ Add an email address

UI - edit


Personal abc@edf.com * trash undo redo delete

csv example not recommended

xml example

```
<email-addresses>
  <email-address>
    <email-address-type></email-address-type>
    <address></address>
  </email-address>
</email-addresses>
```

Notes Email address type is optional. Supported values are:

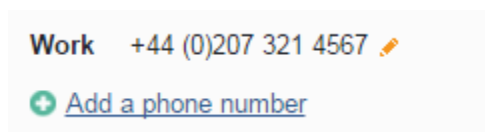
- Personal
- Work
- Other

phone-number-list data type

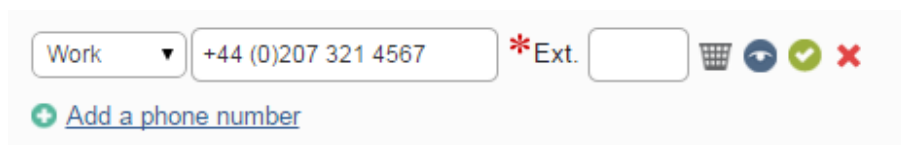
Description A list of phone numbers

Example uses List of phone numbers within a user profile

UI - view



UI - edit



csv example not recommended

xml example

```
<phone-numbers>
  <phone-number>
    <phone-number-type></phone-number-type>
    <number></number>
    <extension></extension>
  </phone-number>
</phone-numbers>
```

Notes Phone number type is optional. Supported values are:

- Work
- Mobile
- Lab
- Fax

