openfunds

White Paper

Version Management

Forward and Backward Compatibility

Today, the transmission of static fund data is still largely a manual process, whereby version management plays a minor role. If a fund distributor requires a new field or an additional attribute in an existing field, it will modify its spreadsheet accordingly and send the fund provider its new requirements. The fund provider will then modify the value manually or semi-automatically and send it to the fund distributor in a spreadsheet or data file.

However, if fund data is to be kept constantly up to date and transmitted automatically, it is necessary to ensure, firstly, that the sending and receiving systems understand each another and, secondly, that the common language—in this case openfunds—can be developed further. openfunds is not alone in facing this challenge: the same applies to virtually all systems that communicate with one another via the Internet. Hyper Text Markup Language (HTML), among the mostly widely recognised languages—maintained by the World Wide Web Consortium and currently in its fifth version—is a case in point. In the same way that a browser must be capable of displaying pages that do not use the latest HTML version, openfunds must be able to guarantee a certain level of backward and forward compatibility.

As far as openfunds is concerned, a receiving system is said to be backward compatible if it accepts data intended for an earlier version of openfunds. For example, the receiver may have already upgraded its database to version 1.1, while the sender only recognises the fields and attributes of version 1.0. If the receiver's database can still read and save the data, then it is said to be backward compatible with version 1.0.

An openfunds database is said to be forward compatible if it is capable of processing data from a more recent version of itself. This is the case, for example, when the receiving database is running on openfunds version 1.1 but is capable of processing data from a database containing openfunds fields from version 1.2. The following diagram illustrates this situation:

Figure 1: Forward and backward compatibility

The database (middle, blue, version 1.1) is backward and forward compatible with data file version 1.0 (left, grey) if it can accept, read and save the data present in this file. It is backward and forward compatible with data file version 1.2 (right, orange) if it can accept, read and save the data present in this file. However, it is very hard to guarantee forward compatibility because when a version 1.1 is created, the future changes that will lead to version 1.2 are unknown. For this reason, openfunds does not guarantee forward compatibility. Backward compatibility is a different matter as the field definitions of version 1.0 are known. If certain rules are observed when creating a version 1.1, backward

compatibility can be guaranteed to a relatively large extent. These rules and the constraints involved are described in the following section.

Backward compatibility in the openfunds Standard

Below are four scenarios of a database in version 1.1, which should be able to read and export the data files of version 1.0 as well as the data files of version 1.1. This can be illustrated schematically as shown in the four cells in the table below: the diagrams differ in terms of the version of the data file to be imported and the version of the data file to be exported:

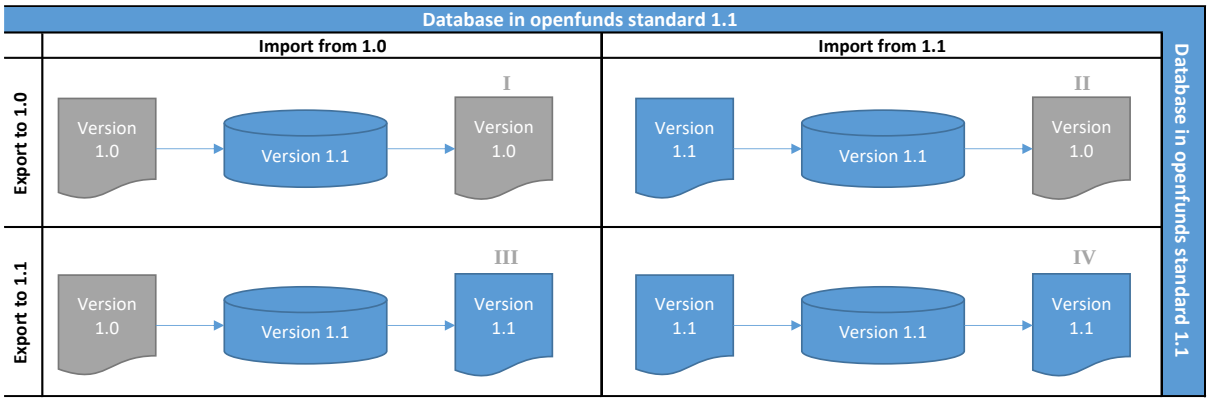

Figure 2: Diagram of an import and export matrix with different openfunds versions

Each of the four cells represents a different scenario. The colours represent the version to be processed: either 1.0 (grey) or 1.1 (blue). In all scenarios, the database that is to read, process and subsequently export the data is in openfunds standard 1.1. The version 1.2 (orange) present in Figure 1 is not shown here as openfunds does not guarantee forward compatibility (see above).

Cell I (top left):
- Data file to be read: Version 1.0
- Data file to be exported: Version 1.0

Cell II (top right):
- Data file to be read: Version 1.1
- Data file to be exported: Version 1.0

Cell III (bottom left):
- Data file to be read: Version 1.0
- Data file to be exported: Version 1.1

Cell IV (bottom right):
- Data file to be read: Version 1.1
- Data file to be exported: Version 1.1

For all the scenarios outlined above, the database is running on openfunds version 1.1

Changes to Fields in Two Successive Versions

In openfunds, version changes involve new, modified and obsolete fields. The six most significant changes affecting fields when a new version is introduced (here: version 1.0 is replaced by version 1.1) are listed below:

1) Version 1.1 contains a new field [N], which was not present in version 1.0.

2) In version 1.1, a field present in version 1.0 has had a value added. (The opposite scenario, i.e. the elimination of values, is deliberately excluded. See the rule at the end of this Article.)

3) In version 1.1, a field that was present in version 1.0 has been deleted and has not been replaced (//No longer supported//).

4) A new field [N] replaces an old field [O]. [N] can be derived from [O] and vice versa.

5) A new field [N] replaces an old field [O]. [N] can be derived from [O] but not vice versa.

6) A new field [N] replaces an old field [O]. [O] can be derived from [N] but not vice versa. If we now insert these six field changes into the matrix above (Figure 2), the following picture emerges:

	Database in openfunds standard 1.1		
	Import from 1.0	**Import from 1.1**	
Export to 1.0	I 1) ✓ 1.0 does not know the field 2) ✓ Value is not transmitted. Therefor no errors. 3) ✓ Date base still knows the no longer supported field 4) ✓ Translation for import and Export, old field deleted 5) ✓ Old and new translated value in DB 6) ✓ Keep old field in DB	II 1) ✓ 1.0 does not know the field 2) (✗) Recieving party receives an unkown value 3) ✗ Version 1.1 does not know the field. Export to 1.0 not possible 4) ✓ Translation for Export 5) ✗ Translation in Version 1.0 not possible, therefor no Export 6) ✓ Translation for Export	Database in openfunds standard 1.1
Export to 1.1	III 1) ✗ DB does not contain any value as the field is missing in V1.0 2) ✓ All improted values can be exported to V1.1 3) ✓ Field is not expected in export 1.1 4) ✓ New value translated from the old value 5) ✓ New value translated from the old value 6) ✗ DB does not contain any value as the field is missing and cannot be derived	IV ✓	

Figure 3: Six possible field changes

Cell I: Importing data fields from version 1.0 and exporting to version 1.0

Scenario 1) Version 1.1 contains a new field [N] which was not present in version 1.0.

This scenario is relatively straightforward as we have a new field [N] here, which the import does not recognise. Therefore, the database will not update the new field [N]. However, as the data is to be exported to version 1.0 as well, and therefore the export does not recognise [N], either, this is irrelevant.

Scenario 2) Version 1.1 contains an old field [O] which has had an attribute added.

This scenario is relatively straightforward, too, as we have a new attribute here in field [N], which is not recognised by the old field [O] or by the import. Therefore, the database will not save this value. As data is to be exported to version 1.0, the receiver will never ask for the new attribute as it does not recognise it.

Scenario 3) In version 1.1, a field [O] which was present in version 1.0 has been deleted and has not been replaced. In the openfunds standard, this is noted accordingly.

This poses no problem either, as long as the old field [O] is saved in the database: if this condition is met, the database is able to read the field [O] during import and to output it again during export to version 1.0. This occurs even though the database itself already runs on version 1.1. Therefore, in this case, it is backward compatible.

Scenario 4) A new field [N] in the database which replaces an old field [O] during data import. [N] can be derived from [O] and vice versa.

This scenario poses no problem insofar as it does not matter whether or not the field is present in version 1.0 or version 1.1. Exporting and importing pose no problem either as the values can be translated in both directions.

Scenario 5) A new field [N] in the database which has replaced an old field [O] during data import. [N] can be derived from [O] but not vice versa (i.e. [O] cannot be derived from [N]).

This scenario is similar to scenario 2). Exporting to both versions is possible provided that both the old field [O] and the new field [N] (the latter possibly translated) are saved.

Scenario 6) A new field [N] in the database which has replaced an old field [O] during data import. [O] can be derived from [N] but not vice versa ([N] cannot be derived from [O]).

In this scenario, the database will not contain a new value [N]. However, since data is to be exported to version 1.0, this poses no problem because during export to version 1.0 only the old value is required. However, this is only possible if the database saves the old field [O].

Cell II: Importing data fields from version 1.1 and exporting to version 1.0

Scenario 1) Version 1.1 contains a new field [N] which is not recognised by version 1.0. The database accepts the new field [N] from version 1.1 and saves it. As the data is exported to version 1.0, the field [N] is not exported, nor is it expected.

Scenario 2) Version 1.1 contains an old field [O] which has had an attribute added.

This is a special case: As the importing database recognises the new attribute the second recipient might receive a value that it cannot validate. This might indicate that attributes have been added: therefore openfunds recommends accounting for this scenario during implementation and adding a note indicating that the field in question needs checking for an update in openfunds.

Scenario 3) In version 1.1, a field [O] from version 1.0 has been deleted and has not been replaced.

As version 1.1 does not recognise the old field [O], the latter cannot be saved in version 1.1. Therefore, an export back to version 1.0 is no longer possible, either.

Scenario 4) A new field [N] in the database which has replaced an old field [O] during data import. [N] can be derived from [O] and vice versa.

This scenario poses no problem insofar as it does not matter whether or not the field is present in version 1.0 or version 1.1. Exporting and importing data pose no problem either as the values can be translated in both directions.

Scenario 5) A new field [N] in the database which has replaced an old field [O] during data import. [N] can be derived from [O] but not vice versa (i.e. [O] cannot be derived from [N]). As in scenario 3), unfortunately, backward compatibility is not provided as [O] can no longer be reconstructed from [N], nor has it been supplied as [O].

Scenario 6) A new field [N] in the database which has replaced an old field [O] during data import. [O] can be derived from [N] but not vice versa (i.e. [N] cannot be derived from [O]). The database contains only the new value [N]. However, as this value can be translated back into the value [O], this poses no problem.

Cell III: Importing data fields from version 1.0 and exporting to version 1.1

Scenario 1) Version 1.1 contains a new field [N] which is not recognised by version 1.0.

The file to be imported (in version 1.0) does not recognise the new field [N]. Therefore, the database has not saved this value and cannot export it.

Scenario 2) Version 1.1 contains an old field [O] which has had an attribute added to it.

The import file version 1.0 may contain values that are recognised by both the database and the export file as both recognise all the values present in version 1.0. Therefore, no data is lost in this scenario.

Scenario 3) In version 1.1, a field [O] from version 1.0 has been deleted and has not been replaced. This fact is noted accordingly in the openfunds standard.

During import, this field is supplied but cannot be saved in the database. Therefore, the value cannot be exported to version 1.1 either. However, since the field is no longer present in version 1.1, this has been marked here with a green "ok".

Scenario 4) A new field [N] in the database which has replaced an old field [O] during data import. [N] can be derived from [O] and vice versa.

This scenario poses no problem insofar as it does not matter whether or not the field is present in version 1.0 or version 1.1. Exporting and importing data pose no problem either as the values can be translated in both directions.

Scenario 5) A new field [N] in the database which has replaced an old field [O] during data import. [N] can be derived from [O] but not vice versa (i.e. [O] cannot be derived from [N]).

This scenario poses no problem as the data is imported into version 1.0 with the old value [O]. This is translated into a new field [N], which is then exported.

Scenario 6) A new field [N] in the database which has replaced an old field [O] during data import. [O] can be derived from [N] but not vice versa (i.e. [N] cannot be derived from [O]).

The import contains only the field [O]. As it is impossible to derive the new field [N] from the old field [O] for the version 1.1, on which the database runs, the database does not contain the value [N]. Therefore, an export is not possible.

Cell IV: Importing data fields from version 1.1 and exporting to version 1.1

This is the most straightforward scenario as all three systems involved (import file, database and export file) run on the latest version. Without iterating the individual scenarios, this is marked with a single green tick.

Major and minor Releases

In a major release, the digit to the left of the decimal point in the version number is incremented by one. For example, a version 1.8 will be followed by version 2.0; In a minor release, the number after the decimal point is incremented. For example, a version 1.2 will be followed by version 1.3.

Rules and principles

openfunds has set itself the following rules and principles with the aim of minimising the cost of implementation, maintenance and development.

- The data ID of an openfunds field cannot be changed
  The data ID is inextricably linked to a field's meaning. In other words, the field name may be changed as long as the field's meaning and data ID remain unchanged. This also means that once a data ID has been assigned, it cannot be used for a different field. This also applies if the field represented by the data ID no longer exists in the current openfunds version.
- No fields disappear from one major release to another
  This means that a field that has become obsolete is simply marked as "//no longer supported…//". In this case, reference is usually made to one or more new fields that are to be used instead.

- **Versioning has to do with fields not files**
  Therefore, a file may contain fields from different versions. To this end, openfunds records the point in time when a field is introduced, and if it is marked as //no longer supported …//, it will record the last release in which the field was supported. An overview is provided in the Excel file under "Fields".

- **Attributes may be added but not removed**
  Attributes (values) of individual fields may not be deleted. If this is required, the field "dies" (//no longer supported …//), and a reference is added to the newly created field. However, attributes may be added as described in case 2) in the matrix above.

- **Major releases will affect backward compatibility**
  One of the reasons for this is because fields marked as //no longer supported … // are dropped, and, consequently, translation may no longer be possible. That said, the assigned data IDs for these //no longer supported … // fields are still blocked. All versions are listed on the openfunds website under "Fields". This ensures that fields that are currently no longer listed can be retrieved by searching for the field list with the highest version number within the previous major releases.

Note - version 1.99

Fields that are in the openfunds database but are not yet assigned to any version for release will typically be assigned the version number 1.99, a test version meaning the field has not been released. Such fields will typically not be visible to users until they are released, but in cases that they are, they should be disregarded as not yet available for use.

Joining openfunds

If your firm has a need to reliably send or receive fund data, you are more than welcome to use the openfunds fields and definitions free-of-charge. Interested parties can contact the openfunds association by sending an email to: businessoffice@openfunds.org

openfunds.org
Staffelstrasse 12
CH-8045 Zürich
Switzerland
Tel.: +41 44 286 80 20
Email: businessoffice@openfunds.org
Website: https://www.openfunds.org

Revision History

Version	Date	Status	Notice
1.1	2020-09-29	Update	Appended note about test version 1.99.
1.0	2018-11-16	Final	First Version.