Project: DRA FRONT END FILTER PROJECT

Title: A Standard ML Specification of the Output Filter

Ref: DS/FMU/FEF/023 *Issue: Revision* : *1.3* *Date:* 5 June 2016

Status: Draft *Type:* Specification

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
G. M. Prout	WIN01		

Authorisation for Issue:

Name	*Function*	*Signature*	*Date*
R.B. Jones	HAT Manager		

Abstract: A specification of the output filter in Standard ML for the DRA front end filter project RSRE 1C/6130.

Distribution: HAT FEF File
Simon Wiseman

0 DOCUMENT CONTROL

0.1 Contents List

0.2 Document Cross References

[1] *The SWORD Front End Filter Specification.* Simon Wiseman, DRA, 21st January 1993.

0.3 Changes History

Issue *Revision* : *1.3* **(5 June 2016)** *Inl* and *Inr* should be *inL* and *inR* (from *fef019*). Incorrect reference cited (should be *filter* not *output*).

Issue 1.4 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document gives a formal specification in Standard ML of the SWORD front end output filter of [1].

1.2 Introduction

2 FILTER FUNCTIONS FOR SELECT QUERIES

SML
```
exception internalError;
```

The type of the data is left as generic. The function $class_{data}$ returns the classification of a piece of data that is to be treated as a classification.

SML
```
fun     (class_data: 'DATA -> Class) d = raise notDefined "class_data";
```

The user's clearance is supplied as parameter to the following functions.

The function *filter_where_row* takes a data list whose first element is the classification of the where clause, removes the classification of the where clause from the head of the list and also returns a boolean which is *true* if the user is not cleared to see the where clause.

SML
```
fun     (filter_where_row : 'DATA list * Class -> 'DATA list * bool)
                            ([],uc) = raise internalError
|       filter_where_row   (d::ds,uc) =  (ds, not (uc dom (class_data d)));
```

The function *filter_where* discards rows where the user is not cleared to see the where clause and also returns a boolean *true* if any rows have been discarded.

SML
```
fun     (filter_where : 'DATA list list * Class -> 'DATA list list * bool)
                    ([],uc) = ([],false)
|       filter_where ((ds::dss),uc) =    let   val(fds,msg) = filter_where_row(ds,uc)
                                              val(fdss,msgs) = filter_where(dss,uc)
                                        in    if msg then (fdss,true)
                                              else (fds::fdss,msgs)
                                        end;
```

If the user is not cleared to see the data in a particular field, the string *not_cleared* is returned.

```
SML
|fun     (filter_cols : Class * ('DATA list * bool list)  -> ('DATA,string)Sum list )
|                      (uc,([],[])) = []
||       filter_cols   (uc,(d::c::ds,true::bs)) =
|                      let     val fd =      if uc dom class_data c
|                                            then inL d
|                                            else inR  "not_cleared"
|                      in fd :: filter_cols(uc,(ds,bs))
|                      end
||       filter_cols   (uc,(d::ds,false::bs)) = inL d :: filter_cols(uc,(ds,bs))
||       filter_cols   other = raise internalError;
```

The boolean parameter to the function *filter_select* is *true* if the lists of data contain the class of the where clause as first elements. The boolean list parameter provides information as to whether it is necessary to check if the user's clearance dominates the classification of the data selected. *filter_select* returns the filtered data together with a boolean which determine whether or not the *mayNotBeComplete* message should be issued.

```
SML
|fun     (filter_select : bool * bool list  * 'DATA list list * Class
|                                              -> ('DATA,string)Sum list list * bool)
|                      (true,cls,dss,uc) =    let val (fdss,nc) = filter_where(dss,uc)
|                                             in ((at2 (map(curry filter_cols uc))
|                                                      (fdss,seq(length fdss,cls))),nc)
|                                             end
||       filter_select (false,cls,dss,uc) =   ((at2 (map(curry filter_cols uc))
|                                                      (dss,seq(length dss,cls))),false);
```

3 INDEX