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

0.1 Contents List

0 DOCUMENT CONTROL ................................................................. 2
   0.1 Contents List ................................................................. 2
   0.2 Document Cross References .................................................. 2
   0.3 Changes History ............................................................... 2
   0.4 Changes Forecast ............................................................. 2

1 GENERAL ................................................................................. 3
   1.1 Scope ............................................................................... 3
   1.2 Introduction ........................................................................ 3

2 FILTER FUNCTIONS FOR SELECT QUERIES ........................................ 3

3 INDEX ....................................................................................... 5

0.2 Document Cross References


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

The type of the data is left as generic. The function \( \text{class}_{\text{data}} \) returns the classification of a piece of data that is to be treated as a classification.

\[
\text{fun \ (class}_{\text{data}} : \ 'DATA \rightarrow \text{Class} \ d = \text{raise notDefined} \ "\text{class}_{\text{data}}";}
\]

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

The function \( \text{filter}_\text{where}_\text{row} \) 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 \( \text{true} \) if the user is not cleared to see the where clause.

\[
\text{fun \ (filter}_\text{where}_\text{row} : \ 'DATA \text{list } \text{Class} \rightarrow \ 'DATA \text{list } \text{bool)}
\]

\[
\quad (\ [], \text{uc}) = \text{raise internalError}
\]

\[
\quad \text{filter}_\text{where}_\text{row} \ (\ d::ds, \text{uc}) = \ (ds, \text{not (uc dom (class}_{\text{data}} d)));
\]

The function \( \text{filter}_\text{where} \) discards rows where the user is not cleared to see the where clause and also returns a boolean \( \text{true} \) if any rows have been discarded.

\[
\text{fun \ (filter}_\text{where} : \ 'DATA \text{list list } \text{Class} \rightarrow \ 'DATA \text{list list } \text{bool)}
\]

\[
\quad (\ [], \text{uc}) = (\ [], \text{false})
\]

\[
\quad \text{filter}_\text{where} \ ((ds::dss), \text{uc}) = \ \text{let}
\]

\[
\quad \ \text{val (fds, msg) = filter}_\text{where}_\text{row}(ds, \text{uc})
\]

\[
\quad \text{val (fdss, msgs) = filter}_\text{where}(dss, \text{uc})
\]

\[
\quad \text{in}
\]

\[
\quad \text{if msg then (fdss, true) else (fds::fdss, msgs)}
\]

\[
\quad \text{end};
\]

If the user is not cleared to see the data in a particular field, the string \( \text{not}\_\text{cleared} \) is returned.
SML

```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

```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))))
    end

| filter_select (false,cls,dss,uc) = ((at2 (map(curry filter_cols uc)))
    (dss,seq(length dss,cls))),false);
```
3 INDEX

class data ........................................ 3
filter_cols ...................................... 4
filter_select .................................... 4
filter_where_row ................................. 3
filter_where ..................................... 3