UNEDA Stakeholder Multi-criteria Layer Interface Specification

Universal Engine for Decision Analysis

Version 7.21

This is the API specification for the SML (Stakeholder Multi-criteria Layer) functional package.

SML is a layer on top of the UNEDA DTL/TDL package, hiding as much of the complexity as possible while at the same time introducing the concept of stakeholders that can be evaluated jointly (weighted or unweighted) or separately. The UNEDA SML API calls are kept much simpler but as similar to DTL as possible to facilitate some limited interoperability. A number of SML calls are extensions to DTL.

The SML commands are divided into eight groups: **System**, **Structure**, **Weights**, **Probabilities**, **Values**, **Evaluation**, **Miscellaneous**, and **Error Handling**.

NOTE: The return codes listed at each function call are the most common ones. For a complete set of return codes, refer to the section on error handling.

CONTENTS

RELEASE HISTORY

Ver.	Date	Main reasons
1.22	220520	SML introduced
1.23	220802	VBA Excel support
1.24	221010	New SML calls
1.26	230325	Error stress test
1.27	230606	New SML CAR calls

DATA TYPES

There are a number of predefined data types in the SML package. These are used for communication between the user layer and SML. Most are based either on *int* or on *double*.

```
typedef double a_vector[MAX_ALTS+1];
typedef a_vector ar_matrix[MAX_ALTS+1];
typedef int ai_vector[MAX_ALTS+1];
typedef ai_vector ai_matrix[MAX_ALTS+1];
typedef double h_vector[MAX_NOPA+1];
typedef h_vector h_matrix[MAX_ALTS+1];
typedef int o_matrix[MAX_ALTS+1][MAX_COPA+1];
typedef double e_matrix[MAX_RESULT+1][MAX_RESULTSTEPS];
typedef int t_row[MAX_NOPA+1];
typedef t_row t_matrix[MAX_ALTS+1];
typedef double ar_col[MAX_ALTS+1];
typedef double cr_col[MAX_CRIT+1];
typedef int ai_col[MAX_ALTS+1];
```

DATA STRUCTURES

The user statements are of two separate types, one for weight statements (user_w_stmt_rec) and the other for probability and value statements (user_stmt_rec).

```
struct user_w_stmt_rec {
      int n_terms;
      int crit[MAX_TERMS+1];
      int sign[MAX_TERMS+1];
      double lobo;
      double upbo;
      };

struct user_stmt_rec {
      int n_terms;
      int alt[MAX_TERMS+1];
      int cons[MAX_TERMS+1];
      int sign[MAX_TERMS+1];
      double lobo;
      double upbo;
      };
```

INDEXING

There are four separate ways of indexing a node or consequence, using either alternative and node number or a node sequence number and using either a total numbering (including intermediate nodes) or a final consequence numbering (excluding intermediate nodes). The numbering is depth-first per alternative in the tree. These four modes (plus two weight modes) are mapped below, and for each command using indexing, the indexing mode is indicated.

Indexing type	Alt. + node	Node sequence	Weight
Total numbering	A1	B1	C1
Final numbering	A2	B2	C2

SYSTEM COMMANDS

Start SML layer

Call syntax: SML_init(mode)
Call syntax: SML_init2(mode)

```
Mode: 0 = V-base source is a human (both calls)
      1 = V-base source is a machine (only init2)
     +2 = stress test of error handling

Return information:
OK    -
ERROR – state error
        frame in use
```

Call semantics: Perform initialisation of SML, CAR, DTL, and TCL resources and start the SML layer. This must be the first call to SML.

Stop SML layer

Call syntax: SML_exit()

```
Return information:
OK    - number of entries written to trace log
ERROR – state error
        frame in use
        memory leak
```

Call semantics: Release resources in SML, CAR, DTL, and TCL. This should be the last call to SML. Check trace log immediately if positive return code.

Abort command

Two versions are available, one for threads or processes sharing addressing space (typically Java callers), the other for interrupt-driven inter-process communication (typically C callers).

Call syntax: SML_abort()

```
Return information:
OK    - user abort queued
```

Call semantics: Must be called by a thread or process sharing address space with the rest of SML. The user request for abort is registered in SML. SML looks for the nearest safe point to stop the calculation. If little remains of the calculation, it will run to the end with the ordinary return code and the call results are valid. If some more remains of the calculation, it will be aborted with the SML_USER_ABORT return code and the call results are then invalid.

Call syntax: send SIGINT signal to the SML process

Return information:
OK - user abort queued

Call semantics: A mechanism for interrupt-driven inter-process communication. The master process sends an interrupt to the slave SML process. The user request for abort is registered in SML. SML looks for the nearest safe point to stop the calculation. If little remains of the calculation, it will run to the end with the ordinary return code and the call results are valid. If some more remains of the calculation, it will be aborted with the SML_USER_ABORT return code and the call results are invalid.

STRUCTURE COMMANDS

Tree structure

Each alternative has its own tree for each criterion. The tree starts with an implicit decision node as node 0 (the root node). The decision tree is expressed as a vector of tree nodes for each alternative. A node is defined as follows:

```
typedef struct tt_node {
    char type;
    int next;
    int down;
    } ttnode;
```

'type' is the node type. Possible types are:
C Consequence node
D Decision node
E Event node

'next' points to the next node at the same level, and 'down' points to the first child of the node (only if the node is an intermediate node of type D or E). The numbering is depth-first. The value zero indicates a null pointer.

Trees are constructed as node vectors, one for each alternative.

```
typedef ttnode ta_tree[MAX_COPA+1];
typedef ta_tree tt_tree[MAX_ALTS+1];
```

Create new frame

There are five types of frames, three *deterministic* (1-3) and two *probabilistic* (4-5):
1) Flat DM-frame with criteria weights, values, and a flat structure (one level, no tree).
2) Tree DM-frame with criteria weights, values, and a weight tree.
3) Tree SM-frame with stakeholders, values, criteria weights, and a weight tree for stakeholders and criteria.
4) Flat PM-frame with probabilities, values, criteria weights, and a flat criteria structure. All criteria have their own event frames.
5) Tree PM-frame with probabilities, values, criteria weights, and a criteria tree. All criteria have their own event frames.

The deterministic types are of two kinds. DM-frames are multi-criteria frames but without event trees, i.e. each criterion has exactly one consequence for each alternative. SM-frames are multi-stakeholder DM-frames, where each stakeholder has a unique set of criteria weights. The probabilistic types consist of a PM-frame containing the multi-criteria weight structure (tree or flat) and slots for holding criteria frames in the form of sub-frames, all handled as a single PM-frame. The sub-frames are independent and possible to evaluate separately in the PM-frame slots. If a slot is unoccupied, a stand-in evaluation of the slot is done for PM-frame evaluations. The stand-in evaluation corresponds to an empty sub-frame.

Call syntax (1): SML_new_DM_flat_frame(int ufnbr, int n_crit, int n_alts)

Return information:
OK -
ERROR - input error
frame unknown
frame exists
too many criteria
too many alternatives

Call semantics: Creates a new deterministic DM-frame with 'n_crit' criteria and 'n_alts' alternatives as specified in the call. Deterministic means that each alternative under each criterion has only one consequence, i.e. no event tree. The frame receives the frame number 'ufnbr'. A frame cannot have less than two alternatives.

Call syntax (2): SML_new_DM_tree_frame(int ufnbr, int n_alts, int n_wtnodes, ta_tree wtree)

Return information:
OK -
ERROR - input error
tree error
frame unknown
frame exists
too many criteria
too many alternatives

Call semantics: Creates a new deterministic DM-tree with as many criteria as there are end nodes in the weight tree as specified in the call, and 'n_alts' alternatives. The weight tree (having 'n_wtnodes' nodes) is supplied in the call and deterministic stubs are created automatically for each criterion. Deterministic means that each alternative under each criterion has only one consequence, i.e. no event tree. The frame receives the frame number 'ufnbr'. A frame cannot have less than two alternatives.

Call syntax (3): SML_new_SM_tree_frame(int ufnbr, int type, int n_alts, int n_sh, int n_nodes, ta_tree wt_tree)

Type: 1 copy stakeholder 1 consequences (same values for all sh)
2 duplicate the stakeholder consequences (different values)

Return information:

```
OK    -
ERROR - input error
        tree error
        frame unknown
        frame exists
        too many stakeholders
        too many criteria
        too many alternatives
```

Call semantics: Creates a new deterministic combined stakeholder-criteria weight tree (having 'n_wtnodes' nodes) with as many stakeholders as specified in 'n_sh' and as many criteria as there are end nodes in the weight tree for a single stakeholder, and 'n_alts' alternatives. The weight tree is supplied in the call and deterministic stubs are created automatically for each stakeholder and criterion. Deterministic means that each alternative under each criterion has only one consequence, i.e. no event tree. The frame receives the frame number 'ufnbr'. A frame cannot have less than two alternatives. NOTE1: The caller supplies the combined stakeholder-criteria tree in the call. It is up to the caller to supply a stakeholder hierarchy in which the lowest level contains the criteria (i.e. the frame has many stakeholder levels but only one criterion level) since this is a mostly stakeholder-focused function. NOTE2: In any data input or evaluation call involving stakeholders, the parameter 'crit' in calls below should be replaced by 'sc(sh,crit)' where sc is a macro taking two parameters: 'sh' is the stakeholder number and 'crit' is the criterion number within the stakeholder. This way, criteria within stakeholders are addressed using the same calls as for single-stakeholder frames.

Call syntax (4): SML_new_PM_flat_frame(int ufnbr, int n_crit, int n_alts)

```
Return information:
OK    -
ERROR - input error
        tree error
        frame unknown
        frame exists
        too many criteria
        too many alternatives
```

Call semantics: Creates a new probabilistic multi-criteria frame with 'n_crit' criteria and 'n_alts' alternatives as specified in the call. The frame receives the frame number 'ufnbr'. A frame cannot have less than two alternatives. The frame is not loaded and can be filled with data prior to loading.

Call syntax (5): SML_new_PM_tree_frame(int ufnbr, int n_alts, int n_wtnodes, ta_tree wtree)

```
Return information:
OK    -
ERROR - input error
        tree error
        frame unknown
        frame exists
        too many criteria
```

```
        too many alternatives
```

Call semantics: Creates a new probabilistic multi-criteria tree frame with as many criteria as there are end nodes in the weight tree as specified in the call. The weight tree is supplied in the call, but the trees for the criteria are supplied in separate calls to SML_new_PM_crit_tree or SML_load_PM_crit. A frame cannot have less than two alternatives. The weight tree must have at least one node. 'n_wtnodes' does not include the root node. The frame is not loaded and can be filled with data prior to loading. The weight tree is specified for each alternative as node pointers 'next' and 'down' for each node. 'next' points to the next node at the same level, and 'down' points to the children of the node (only if the node is an intermediate node). The value 0 indicates a null pointer.

Create new criterion

Call syntax: SML_new_PM_crit_tree(int crit, int n_nodes[], tt_tree xtree)

```
Return information:
OK    -
ERROR - input error
        tree error
        criterion exists
        criterion unknown
        wrong frame type
        too many consequences
```

Call semantics: Creates a new criterion 'crit' with a tree as specified in the call. The criterion is added to the loaded PM-frame. The tree is specified for each alternative as node pointers 'next' and 'down' for each node. 'next' points to the next node at the same level, and 'down' points to the children of the node (only if the node is an intermediate node). The value zero indicates a null pointer.

Delete a criterion

Call syntax: SML_delete_PM_crit(int crit)

```
Return information:
OK    - frame number
ERROR - criterion unknown
        frame not loaded
        wrong frame type
```

Call semantics: Deletes the criterion in the slot 'crit' from the loaded PM-frame. The criterion cannot subsequently be recovered into a PS-frame.

Check frame type

Call syntax: SML_frame_type(int ufnbr)

```
Return information:
OK    - frame type: 1=SM1 (same values), 2=SM2 (diff values), 3=DM/PM
ERROR – frame unknown (ufnbr out of range)
```

Call semantics: Checks the type of the frame 'ufnbr'. Supplying 0 as 'ufnbr' indicates the currently loaded frame. Returns the frame type if the frame number is associated with a user frame in SML and 0 otherwise.

Check criterion

Call syntax: SML_PM_crit_exists(int crit, int *exists)

Return information:
OK -
ERROR – criterion unknown
frame not loaded
wrong frame type

Call semantics: Checks if the criterion exists. Returns TRUE in 'exists' if the criterion slot number 'crit' is associated with a frame and FALSE otherwise.

Dispose of frame

Call syntax: SML_dispose_frame(int ufnbr)

Return information:
OK -
ERROR – frame in use
frame unknown

Call semantics: Dispose of resources belonging to frame 'ufnbr' and free the position for a new frame. NOTE: Frames can only be disposed of when no frame is open.

Load frame

Call syntax: SML_load_frame(int ufnbr)

Return information:
OK - for PM-frames: number of connected probability trees
ERROR – frame unknown
frame corrupted
frame in use
inconsistent

Call semantics: Attempts to attach the frame 'ufnbr' to DTL. Bases are loaded and checked for consistency. If any base is inconsistent, the frame will not be attached (loaded).

Close frame

Call syntax: SML_unload_frame()

Return information:
OK -

ERROR - frame not loaded

Call semantics: Detach the frame from DTL/TCL and free the interface for new frames. NOTE: In case of internal problems in DTL/TCL, the frame might be detached without an explicit call to SML_unload_frame.

FILE COMMANDS

Read frame from file

Call syntax: SML_read_frame(int ufnbr, int type, int n_sh, char *fn, char *folder)

Return information:
OK -
ERROR – file corrupt
file/folder unknown
frame exists

Call semantics: Reads the file 'fn' of type 'type' in folder 'folder' and creates a user frame 'ufnbr' from the file. The file should have been previously written by SML_write_frame and contain 'n_sh' stakeholders.

Write frame to file

Call syntax: SML_write_frame(char *fn, char *folder)

Return information:
OK -
ERROR – frame not loaded
frame corrupt

Call semantics: Writes the currently loaded user frame to the file 'fn' in folder 'folder'.

WEIGHT COMMANDS

Weights can be criteria weights, stakeholder weights, or similar. All kinds of weights in the weight hierarchy (tree) are treated in the same way within the two node categories: intermediate nodes and final (real) nodes.

Set weight base

Call syntax: SML_set_W_base(h_vector lobox, h_vector mbox, h_vector upbox)
Call syntax: SML_set_W_base2(h_vector lobox, h_vector mbox, h_vector upbox, int *inc_var)

Return information:
OK -
ERROR - wrong frame type
frame not loaded
inconsistent

Call semantics: Range statements for all criteria weights (in three vectors 'lobox', 'mbox', and 'upbox' indexed as [node]) are added to the weight base at the same time. An inactive entry in 'mbox' is marked with -1.0. The base is checked for consistency with respect to all new ranges. In case of inconsistency, nothing is added to the base and for version 2 of the call the offending variable number is returned in 'inc_var' if known. Indexing type: C1. NOTE: 'node' is the node number in the weight tree.

Get weight hull

Call syntax: SML_get_W_hull(int global, h_vector lobo, h_vector mid, h_vector upbo)

Return information:
OK -
ERROR - wrong frame type
frame not loaded
too many consequences

Call semantics: The global ('global'=TRUE) or local ('global'=FALSE) hull and midpoint are returned in three vectors 'lobo', 'mid', and 'upbo' indexed as [node]. Indexing type: C1. NOTE: 'node' is the node number in the weight tree.

PROBABILITY COMMANDS

Set probability base

Call syntax: SML_set_P_base(int crit, h_matrix lobox, h_matrix mbox, h_matrix upbox)
Call syntax: SML_set_P_base2(int crit, h_matrix lobox, h_matrix mbox, h_matrix upbox, int *inc_var)

Return information:
OK -
ERROR - wrong frame type
criterion unknown
frame not loaded
inconsistent

Call semantics: Range statements for all consequences (in three matrices 'lobox', 'mbox', and 'upbox' indexed as [alt][node]) are added to the probability base of the criterion 'crit' at the same time. An inactive entry in 'mbox' is marked -1.0. The base is checked for consistency with respect to all new ranges. In case of inconsistency, nothing is added to the base and for version 2 of the call the offending variable number is returned in 'inc_var' if known. Indexing type: A1. NOTE: 'lobox' and 'upbox' must contain local probabilities.

Get probability hull

Call syntax: SML_get_P_hull(int crit, int global, h_matrix lobo, h_matrix mid, h_matrix upbo)

```
Return information:
OK    -
ERROR - wrong frame type
        criterion unknown
        frame not loaded
        too many consequences
```

Call semantics: The global ('global'=1) or local ('global'=0) hull and midpoint of the criterion 'crit' are returned in three matrices 'lobo', 'mid', and 'upbo' indexed as [alt][node]. Indexing type: A1.

VALUE COMMANDS

Set value base

Call syntax: SML_set_V_base(int crit, int rev, int renorm, h_matrix lobox, h_matrix mbox, h_matrix upbox)
Call syntax: SML_set_V_base2(int crit, int rev, int renorm, h_matrix lobox, h_matrix mbox, h_matrix upbox, int *inc_var)

```
Rev:    0 = standard scale, higher values are preferred
        1 = reverse scale, lower values are preferred

Renorm: 0 = do not renormalise value base scale
        1 = renormalise value base scale
        2 = automatic, SML handles renormalisation

Return information:
OK    -
ERROR - wrong frame type
        criterion unknown
        frame not loaded
        inconsistent
```

Call semantics: Range and mbox (most likely) statements for all consequences (in three matrices 'lobox', 'mbox', and 'upbox' indexed as [alt][node]) are added to the value base of the criterion 'crit' at the same time. In case of inconsistency, nothing is added to the base and the scale is not changed. For version 2 of the call the offending variable number is returned in 'inc_var'. Indexing type: A1. NOTE: Since values do not sum to one (or any other fixed number), there is no option to omit the mbox entries.

Get value hull

Call syntax: SML_get_V_hull(int crit, h_matrix lobo, h_matrix mid, h_matrix upbo)

```
Return information:
OK    -
ERROR - frame not loaded
        criterion unknown
        too many consequences
```

Call semantics: The hull and the midpoint of the criterion 'crit' are returned in three matrices 'lobo', 'mid', and 'upbo' indexed as [alt][node]. Indexing type: A1.

EVALUATION COMMANDS

For most evaluation commands, multi-criteria (PM/DM-frame) or multi-stakeholder (SM-frame) evaluations are invoked by supplying crit=0. Partial evaluations of the weight tree (stakeholder and/or criteria) can be invoked by crit<0, where |crit| is the node number to start at. It must be an intermediate node. For end nodes, use a positive crit argument.

Command	Crit>0	Crit=0	Crit<0
SML_evaluate_frame	x	x	x
SML_compare_alternatives	x	x	x
SML_delta_mass	x	x	x
SML_rank_alternatives	x	x	x
SML_daisy_chain	x	x	x
SML_pie_chart	x	x	x
SML_get_W_tornado			
SML_get_P_tornado	x		
SML_get_MCP_tornado	x		
SML_get_V_tornado	x		
SML_get_MCV_tornado	x		
SML_get_cons_influence	x		
SML_get_mass_range	x	x	x
SML_get_mass_above	x	x	x
SML_get_mass_below	x	x	x
SML_get_support_mass	x	x	x
SML_get_support_lower	x	x	x
SML_get_support_upper	x	x	x

Set multi-criteria scale

Call syntax: SML_set_scale(double v_min, double v_max)

```
Return information:
OK    -
ERROR - wrong frame type
        frame not loaded
        input error
```

Call semantics: Sets the endpoints of the multi-criteria scale. To have lower values being preferred (reverse scale), enter v_min larger than v_max. NOTE: Only the MC scale is allowed to be set manually, otherwise the meaning of value statements would change.

Copy multi-criteria scale

Call syntax: SML_copy_scale(int crit)

```
Return information:
OK    -
ERROR - wrong frame type
        frame not loaded
        criterion unknown
```

Call semantics: Copies the endpoints of the scale of the criterion 'crit' specified in the call onto the multi-criteria scale. This call equalises the two scales' endpoints.

Check scale values

Call syntax: SML_check_user_values(int crit, int type, int count, ...)
Call syntax: SML_check_norm_values(int type, int count, ...)

```
Return information:
OK    -
ERROR - input error
        frame not loaded
        criterion unknown
```

Call semantics: Check that the supplied list of 'count' values (max 10, in separate arguments) are within the scale range. 'type' is absolute ABS_SCALE, difference DIFF_SCALE, or distance DIST_SCALE. This is a variadic function call which accepts a varying number of arguments (indicated by the ellipsis).

Evaluate frame

Call syntax: SML_evaluate_frame(int crit, int method, int Ai, int Aj, e_matrix e_result)

```
Method subfield:
Eval:    0 DELTA
         4 GAMMA
         8 PSI
        12 DIGAMMA

Return information:
OK    -
ERROR - input error
        criterion unknown
        alternative unknown
        wrong method
        frame not loaded
        output error
```

Call semantics: Evaluate the criterion 'crit' of the loaded frame. All alternatives are evaluated using the Delta, Gamma, Psi, or Digamma rule. For the requested alternative(s) 'Ai' (and 'Aj'), the result is stored in 'e_result'. Each result has the form of a matrix {min,mid,max} x {mass steps}, with values from increasing mass. 'Aj' is relevant only for Delta and Digamma evaluations. For Digamma, 'Aj' contains a bitmap with the selected alternatives starting with alternative 1 in the lowest bit of the map.

Evaluate CDF

Call syntax: SML_evaluate_cdf(int crit, int Ai, c_vector level, c_vector cdf)

```
Return information:
OK    -
ERROR - input error
        criterion unknown
        alternative unknown
        frame not loaded
        output error
```

Call semantics: Evaluate the criterion 'crit' of the loaded frame using the Psi rule. For the requested alternative 'Ai', the EV level and cumulative density function (CDF) are stored in 'level' and 'cdf' respectively.

Evaluate all criteria

Call syntax: SML_evaluate_mid(int Ai, int mode, cr_col o_result, ci_col o_rank)

Evaluate the alternative 'Ai' of the loaded frame w.r.t. all criteria one at a time.

```
Mode:    0 Ordering
         1 Ranking (olympic)
        +2 Output in percent of MC scale
        +4 Renormalisation

Return information:
OK    -
ERROR - input error
        alternative unknown
        frame not loaded
        wrong frame type
```

Call semantics: An alternative is evaluated in each criterion by the Omega rule ("part worth"). The result is stored in 'o_result' indexed with criterion number and the rank or order in 'o_rank'. 'mode' is 0 for ordering (o_rank[i] contains the index of the criterion ranked in position i) and 1 for ranking (o_rank[i] contains the rank position for criterion i). o_result[0] contains the full Omega value for alternative 'Ai' (coinciding with mid for Psi evaluation). If 'Ai' is 0, an average of all alternatives is returned.

Evaluate all criteria at first tree level

Call syntax: SML_evaluate_omega(int Ai, cr_col o_result)
Call syntax: SML_evaluate_omega1(int Ai, cr_col o_result, ci_col o_node)
Call syntax: SML_evaluate_omega2(int Ai, int mode, cr_col o_result, ci_col o_node)

Evaluate the alternative 'Ai' of the loaded frame w.r.t. the total contribution from each node at the first weight tree level.

```
Mode:    0 Output in absolute MC scale
         2 Output in percent of MC scale
         4 Renormalisation

Return information:
OK    -
ERROR - input error
        alternative unknown
        frame not loaded
        wrong frame type
```

Call semantics: An alternative is evaluated in each node at the first weight tree level by the Omega rule ("part worth"). The result is stored in 'o_result' indexed with node number in 'o_node'. o_result[0] contains the full Omega value for alternative 'Ai'. If 'Ai' is 0, an average of all alternatives is returned. NOTE: Only applicable to single stakeholder weight trees.

Consequence influence

Call syntax: SML_get_cons_influence(int crit, int mode, h_matrix result)

```
Mode:    0 Local EV contribution
         1 Global WEV contribution

Return information:
OK    -
ERROR - frame not loaded
        input error
        criterion unknown
```

Call semantics: The influence of the consequences of the criterion 'crit' is returned in the matrix 'result' indexed as [alt][node]. 'mode' is 0 for a local result (i.e. within the criterion) and 1 for a global result (i.e. contribution from the criterion to the weighted expected value). For each final consequence node, the value shows how much the mass point of this particular consequence influences the (weighted) expected value. Indexing type: A1. NOTE: 'node' is the node number in the weight tree.

Compare alternatives

Call syntax: SML_compare_alternatives(int crit, int method, double belief_level, ar_col lo_value, ar_col up_value)

```
Method subfield:
Eval:    4 GAMMA
         8 PSI

Return information:
OK    -
ERROR - frame not loaded
        input error
        criterion unknown
```

Call semantics: Compares alternatives based on 'method' for the criterion 'crit'. The comparison is made using belief mass. The desired belief level in the range [0,1] must reside in 'belief_level' when calling the function. The result is a support range [lo_value[Ai],up_value[Ai]] for each alternative Ai (from 1 to n_alts).

Mass delta between alternatives

Call syntax: SML_delta_mass(int crit, ar_matrix delta_mass, ai_col delta_order)
Call syntax: SML_delta_mass2(int crit, int mode, ar_matrix delta_mass, ai_col delta_order)

```
Mode:   0 no interpolation
        1 interpolation: no mass matrix row may decrease (default)

Return information:
OK    -
ERROR - frame not loaded
        input error
        criterion unknown
```

Call semantics: Returns a matrix 'delta_mass' with the cdf mass of the deltas (differences) in belief mass between each pair [Ai,Aj] of alternatives. 'delta_order' is the alternative numbers in ranking order.

Rank alternatives

Call syntax: SML_rank_alternatives(int crit, int mode, double gamma_tolerance, double omega_tolerance, ai_col gamma_rank, ai_col omega_rank, ar_col gamma_value, ar_col omega_value)
Call syntax: SML_rank_gamma(int crit, ai_col gamma_rank, ar_col gamma_value)
Call syntax: SML_rank_omega(int crit, ai_col omega_rank, ar_col omega_value)

```
Mode:   0 olympic ranking
        1 hard/strict ranking (default for SML_rank_gamma/omega)
       +2 tolerances are values (default: percent)

Return information:
OK    - ok
        differing ranks
ERROR - frame not loaded
        input error
        criterion unknown
```

Call semantics: Obtains the ordinal and cardinal rankings (from 1 to n_alts) of all alternatives based on (i) Omega values (mass points) and/or on (ii) the Gamma evaluations for the criterion 'crit'. The cardinal ranking vectors that the ordinal rankings (range: [1..n]) are based on are returned. It returns SML_DIFFERING_RANKS if the two ordinal ranking vectors are not identical. The closeness tolerances must be in the range [0%,10%] (0% for sharp ordinal ranking) and corresponding for values ('mode'+2).

Daisy chain

Call syntax: SML_daisy_chain(int crit, ai_col daisy_rank, ar_col daisy_value)
Call syntax: SML_daisy_chain2(int crit, int mode, ai_col omega_rank, ar_col daisy_value, ar_col omega_value)

Mode: 0 return absolute omega EV values
1 return relative omega EV values (default)

Return information:
OK -
ERROR - frame not loaded
input error
criterion unknown

Call semantics: Obtains the ordinal and daisy chain (dominance-based) rankings (from 1 to n_alts) of all alternatives based on (i) Omega values (mass points) and on (ii) the pairwise dominance of the alternatives as ranked by the Omega function.

Pie chart

Call syntax: SML_pie_chart(int crit, ar_col pie_value)
Call syntax: SML_pie_chart1(int crit, ar_col pie_value)
Call syntax: SML_pie_chart2(int crit, double moderation, ar_col pie_value)

Negative moderation modifies only the starting point (anchor) of the pie chart. It controls how much of its mass the best alternative distributes along the daisy chain. 0.0 means keep all (default), -1.0 is maximum effect. Positive moderation modifies both the anchor but also the daisy chain as a basis for the chart. Thus, it also controls how much of their mass the other alternatives distribute along the daisy chain. 1.0 is maximum effect.

Return information:
OK -
ERROR - frame not loaded
input error
criterion unknown

Call semantics Obtains the rating of all alternatives based on the mass distribution of Gamma evaluations for the criterion 'crit'. The rating is relative (proportional) intended for e.g. pie charts. The elements in the rating sum up to 100%. The rating vector (range: [0,1]) is returned. SML_pie_chart has no moderation (raw mode), SML_pie_chart1 has low moderation while SML_pie_chart2 gives control over the moderation.

Remaining mass at result level

Call syntax: SML_get_mass_above(double lo_level, double *mass)
Call syntax: SML_get_mass_below(double up_level, double *mass)
Call syntax: SML_get_mass_range(double lo_level, double up_level, double *mass)

Return information:
OK -

```
ERROR - output error
        input error
        criterion unknown
```

A note on belief mass functions

```
Let a,b,c be real numbers in [0,1]
Let s be the lower endpoint of the scale [0,1] (psi) or [-1,1] (delta, gamma)
Let d,e,p be real numbers (points) on the scale [s,1]
Let I(d,e) f(x)dx be the Lebesgue integral from d to e over f(x)
Let dens(x) be a belief density function with I(s,1) dens(x)dx = 1

In theory, the most natural would be a three-way belief function:
a = Belief in interval below point p = I(s,p) dens(x)dx
b = Belief in the point p itself     = I(p,p) dens(x)dx
c = Belief in interval above point p = I(p,1) dens(x)dx

For normal density:
b = 0
a + c = 1
a + b + c = 1

For Dirac density:
b = 1
a + c = 0
a + b + c = 1

But the most efficient implementation is a two-way function:
a = Belief in interval at and below point p
c = Belief in interval at and above point p

For normal density:
a + c = 1

For Dirac density (not at scale endpoints):
a = c = 1/2

For Dirac density (at scale lower endpoint = s):
a = 0
c = 1

For Dirac density (at scale upper endpoint = 1):
a = 1
c = 0

The two-way implementation works perfectly for normal cases but requires special
attention for pointwise masses.

The underlying function does not know whether it is being called by a function
having s=-1 or s=0, so it will return the following:

For Dirac density (at Delta/Gamma/Digamma scale lower endpoint s=-1):
a = 0
c = 1

For Dirac density (at Psi scale lower endpoint s=0):
a = c = 1/2
```

Call semantics: Obtains the fraction [0,1] of the mass remaining above/below a specific result level in the evaluation result of the latest evaluation (or

between the given levels in case of SML_get_mass_range. The fraction is returned in 'mass'. The call must be preceded by an evaluation. This can be seen as the remaining mass above or below a specified result level (or both for SML_get_mass_range) in a traditional evaluation. In that sense, it works perpendicular to the other mass calls.

Support level mass

Call syntax: SML_get_support_mass(double belief_level, double *lobo, double *upbo)
Call syntax: SML_get_support_lower(double belief_level, double *lobo, double *upbo)
Call syntax: SML_get_support_upper(double belief_level, double *lobo, double *upbo)

Return information:
OK -
ERROR - output error
input error
frame not loaded
criterion unknown

Call semantics: Obtains the interval [0,1] within which 'belief_level' fraction of the remaining mass resides in the evaluation result of the criterion 'crit'. 'belief_level' must be in the range [0.5,0.999]. The calculations are the result of a B-normal evaluation. The interval is returned as [lobo,upbo]. The call must be preceded by an evaluation.

Weight tornado

Call syntax: SML_get_W_tornado(h_matrix t_lobo, h_matrix t_upbo)
Call syntax: SML_get_W_tornado2(int mode, h_matrix t_lobo, h_matrix t_upbo)

Mode subfield:
Type: 0 Standard evaluation, explicit midpoint kept (default)
1 Explicit midpoint removed before calculations
+2 Belief mass-based instead of expected value-based

Return information:
OK -
ERROR - frame not loaded
input error
wrong frame type

Call semantics: The weight sensitivity tornado of all alternatives is returned in two matrices (first call) or vectors (second call) 't_lobo' and 't_upbo'. 'mode' 0 is with the explicit midpoint kept and 1 is without an explicit midpoint. For each node, the [t_lobo,t_upbo] interval shows how much the midpoint shifts when the respective weights are set to their minima and maxima one at a time. Indexing type: A1. NOTE: 'node' is the node number in the weight tree.

Probability tornado

Call syntax: SML_get_P_tornado(int crit, h_matrix t_lobo, h_matrix t_upbo)
Call syntax: SML_get_P_tornado2(int crit, int mode, h_matrix t_lobo, h_matrix t_upbo)

```
Mode subfield:
Type:    0 Standard evaluation, explicit midpoint kept (default)
         1 Explicit midpoint removed before calculations
        +2 Belief mass-based instead of expected value-based

Return information:
OK    -
ERROR - frame not loaded
        input error
        criterion unknown
```

Call semantics: The probability sensitivity tornado of the criterion 'crit' is returned in two matrices 't_lobo' and 't_upbo' indexed as [alt][node]. 'mode' 0 is with the explicit midpoint kept and 1 is without an explicit midpoint. Adding 2 to 'mode' yields belief mass-based evaluation instead of expected value-based which takes some more CPU power. For each node, the [t_lobo,t_upbo] interval shows how much the midpoint shifts when the respective probabilities are set to their minima and maxima one at a time. Indexing type: A1.

Criteria probability tornado

Call syntax: SML_get_MCP_tornado(int crit, h_matrix t_lobo, h_matrix t_upbo)
Call syntax: SML_get_MCP_tornado2(int crit, int mode, h_matrix t_lobo, h_matrix t_upbo)

```
Mode subfield:
Type:    0 Standard evaluation, explicit midpoint kept (default)
         1 Explicit midpoint removed before calculations
        +2 Belief mass-based instead of expected value-based

Return information:
OK    -
ERROR - frame not loaded
        input error
        wrong frame type
        criterion unknown
```

Call semantics: The criterion weighted probability tornado of the criterion 'crit' is returned in two matrices 't_lobo' and 't_upbo' indexed as [alt][node]. 'mode' 0 is with the explicit midpoint kept and 1 is without an explicit midpoint. Adding 2 to 'mode' yields belief mass-based evaluation instead of expected value-based which takes some more CPU power. For each final consequence node, the [t_lobo,t_upbo] interval shows how much the midpoint shifts when the respective values are set to their minima and maxima one at a time and how much this influences the total weighted expected value. Indexing type: A1.

Value tornado

Call syntax: SML_get_V_tornado(int crit, h_matrix t_lobo, h_matrix t_upbo)
Call syntax: SML_get_V_tornado2(int crit, int mode, h_matrix t_lobo, h_matrix t_upbo)

```
Mode subfield:
Type:    0 Standard evaluation, explicit midpoint kept (default)
         1 Explicit midpoint removed before calculations
        +2 Belief mass-based instead of expected value-based

Return information:
OK    -
ERROR - frame not loaded
        input error
        criterion unknown
```

Call semantics: The value sensitivity tornado of the criterion 'crit' is returned in two matrices 't_lobo' and 't_upbo' indexed as [alt][node]. 'mode' 0 is with the explicit midpoint kept and 1 is without an explicit midpoint. Adding 2 to 'mode' yields belief mass-based evaluation instead of expected value-based which takes some more CPU power. For each final consequence node, the [t_lobo,t_upbo] interval shows how much the midpoint shifts when the respective values are set to their minima and maxima one at a time. Indexing type: A1.

Criteria value tornado

Call syntax: SML_get_MCV_tornado(int crit, h_matrix t_lobo, h_matrix t_upbo)
Call syntax: SML_get_MCV_tornado2(int crit, int mode, h_matrix t_lobo, h_matrix t_upbo)

```
Mode subfield:
Type:    0 Standard evaluation, explicit midpoint kept (default)
         1 Explicit midpoint removed before calculations
        +2 Belief mass-based instead of expected value-based

Return information:
OK    -
ERROR - frame not loaded
        input error
        wrong frame type
        criterion unknown
```

Call semantics: The criterion weighted value tornado of the criterion 'crit' is returned in two matrices 't_lobo' and 't_upbo' indexed as [alt][node]. 'mode' 0 is with the explicit midpoint kept and 1 is without an explicit midpoint. Adding 2 to 'mode' yields belief mass-based evaluation instead of expected value-based which takes some more CPU power. For each final consequence node, the [t_lobo,t_upbo] interval shows how much the midpoint shifts when the respective values are set to their minima and maxima one at a time and how much this influences the total weighted expected value. Indexing type: A1.

MISCELLANEOUS COMMANDS

Library release version

Call syntax: SML_get_release(string(relstrg))

Return information:

OK -

Call semantics: Obtains the release version of the underlying DTL package. The format for the standard version is **"M.F.T"**, where **M**=main, **F**=functional, and **T**=technical version numbers. The user interface code must assure that **M** and **F** match the application requirements. The standard and long versions both uniquely identify the software library.

Number of weights

Call syntax: SML_nbr_of_weights()

Return information:
OK - number of weight nodes in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of weight nodes in the currently loaded frame. For a frame with several stakeholders, this includes all stakeholders.

Number of criteria

Call syntax: SML_nbr_of_crit()

Return information:
OK - number of criteria in the current frame
ERROR - frame not loaded

Call semantics: Returns the total number of criteria in the currently loaded frame. For a frame with several stakeholders, this includes all stakeholders.

Number of alternatives

Call syntax: SML_nbr_of_alts()

Return information:
OK - number of alternatives in the current frame
ERROR - frame not loaded

Call semantics: Returns the number of alternatives in the currently loaded frame.

Total number of consequences

Call syntax: SML_total_cons(int crit)

Return information:
OK - number of consequences in the specified alternative
ERROR - frame not loaded
criterion unknown

Call semantics: Returns the total number of consequences in all alternatives of the criterion ‘crit’ in the currently loaded frame. For ‘crit’=0, the total number of consequences in the frame is returned. Indexing type: B2.

Number of consequences

Call syntax: SML_nbr_of_cons(int crit, int alt)

Return information:
OK - number of consequences in the specified alternative
ERROR - frame not loaded
criterion unknown
alternative unknown

Call semantics: Returns the number of consequences in the specified alternative of the criterion 'crit' in the currently loaded frame. Indexing type: B2.

Total number of nodes

Call syntax: SML_total_nodes(int crit)

Return information:
OK - number of nodes in all alternatives in total
ERROR - frame not loaded
criterion unknown

Call semantics: Returns the total number of nodes in all alternatives of the criterion 'crit' in the currently loaded frame. For 'crit'=0, the total number of nodes in the frame is returned. Indexing type: B1.

Number of nodes

Call syntax: SML_nbr_of_nodes(int crit, int alt)

Return information:
OK - number of nodes in the specified alternative
ERROR - frame not loaded
criterion unknown
alternative unknown

Call semantics: Returns the number of nodes in the specified alternative of the criterion 'crit' in the currently loaded frame. Indexing type: B1.

Criterion index number

Call syntax: SML_crit_nbr(int sh, int crit)

Return information:
OK - index number in weight tree
ERROR – 0 (input error)

Call semantics: Returns the index number in the weight tree for criterion 'crit' under stakeholder 'sh' (or 0 if input error). Indexing type: C2.

Stakeholder node check

Call syntax: SML_is_stakeholder(int node)

Return information:
OK - TRUE/FALSE
ERROR - frame not loaded
wrong frame type
input error

Call semantics: Returns TRUE if the index in 'node' is a stakeholder node and FALSE if it is a criterion node. Indexing type: C1.

ERROR HANDLING

All SML calls (except SML_get_errtxt) return a number of type *rcode* which serves as an information carrier and error code at the same time. In the event of an error, a negative number is returned. The caller should interpret the error code and take action accordingly. The codes are found below.

Get error text

Call syntax: char *SML_get_errtxt(int drc)
Call syntax: char *SML_get_errtxt2(int drc, int style)

Return information:
OK - pointer to error text
ERROR – pointer to text "- NO TEXT -"

Call semantics: Returns the text string that corresponds to the supplied SML error number in C-style (null-terminated, 'style'=0) or Pascal shortstring-style (length-preceded, 'style'=1) format (automatic for SML_get_errtxt). If the number is out of range, the error text "- NO TEXT -" is returned.

Check error code

Call syntax: SML_error(int drc)

Return information:
0 – the return code 'drc' contains only information
1 - the return code 'drc' contains an error

Call semantics: Returns the severity of the return code 'drc' supplied. The 'drc' code should originate from a previous SML call. The function takes care of both SML and DTL/TCL error codes.

Call syntax: SML_error2(int drc)

Return information:
0 – the return code 'drc' contains information, output valid
1 – the return code 'drc' contains information, output invalid
2 - the return code 'drc' contains an error

Call semantics: Returns the severity of the return code ‘drc’ supplied. The ‘drc’ code should originate from a previous SML call. The function takes care of both SML and DTL/TCL error codes and categorises them as severe (2) or not (1).

Check user-caused error code

Call syntax: SML_u_error(int drc)

Return information:
0 – the return code ‘drc’ contains information or user mistake
1 - the return code ‘drc’ contains an error not caused by a user

Call semantics: Returns the severity of the return code ‘drc’ supplied. The ‘drc’ code should originate from a previous SML call. The function takes care of both SML and DTL/TCL error codes.

Call syntax: SML_u_error2(int drc)

Return information:
0 – the return code ‘drc’ contains information, output valid
1 – the return code ‘drc’ contains information or user mistake, output invalid
2 - the return code ‘drc’ contains an error

Call semantics: Returns the severity of the return code ‘drc’ supplied. The ‘drc’ code should originate from a previous SML call. The function takes care of both SML and DTL/TCL error codes and categorises them as severe (2) or not (1).

Inline error code check

Call syntax: scall(SML_function(par1,par2,...))
Call syntax: ucall(SML_function(par1,par2,...))

Return information:
< -2: DTL return code = real error
  -2: no result but not real error
  -1: inconsistent user input (only ucall)
   0: ok
   1: ok + additional state information
 > 1: ok + additional numeric information

Call semantics: Generalised error code interpreter and handler. Fetches the appropriate error code and combines it with the return code into one single error number.

SML error codes

SML_KERNEL_ERROR

The error occurred in the underlying TCL calculation kernel layer. This value is not returned alone but instead added to the TCL error code.

SML_INPUT_ERROR

One of the input parameters contained invalid information.

SML_TREE_ERROR

The tree structure supplied is invalid or the tree description contained invalid information.

SML_OUTPUT_ERROR

The requested output from the SML function could not be generated. This usually refers to a request for impossible evaluation data.

SML_FRAME_EXISTS

The frame number already exists. No more frames can have the same number.

SML_FRAME_UNKNOWN

The requested frame number does not exist. Either it is not created, or the number is out of range.

SML_FRAME_IN_USE

An attempt to delete or in another way eliminate a frame that is currently attached (loaded).

SML_FRAME_NOT_LOADED

An attempt to use frame commands while no frame is loaded.

SML_FRAME_CORRUPT

Internal error. The frame has been rendered corrupt, either by modifications outside of TCL or because of an internal error in TCL.

SML_WRONG_FRAME_TYPE

An attempt to issue a PS/PM-only command to a DM frame or vice versa.

SML_WRONG_STATEMENT_TYPE

The user statement passed in the call is inappropriate for the type of frame currently loaded.

SML_CONS_OVERFLOW

Too many consequences in the problem for SML to handle. This should be prohibited in the user interface at an earlier point (use MAX_CONS).

SML_CRIT_OVERFLOW

Too many criteria in the problem for SML to handle. This should be prohibited in the user interface at an earlier point (use MAX_CRIT).

SML_ALT_OVERFLOW

Too many alternatives in the problem for SML to handle. This should be prohibited in the user interface at an earlier point (use MAX_ALT).

SML_NODE_OVERFLOW

Too many nodes in the tree for SML to handle. This should be prohibited in the user interface at an earlier point (use MAX_NODES).

SML_DIFFERING_RANKS

The rankings obtained with Omega values (midpoint) and Gamma values are not the same. The results are correct but not in accordance with each other.

SML_SYS_CORRUPT

The internal data structures of SML or DTL are misaligned.

SML_STATE_ERROR

A call to SML is made when SML is in the wrong initialisation state.

SML_CRIT_UNKNOWN

The requested criterion does not exist. The criterion number is within the valid range, but no criterion has been installed at this position.

SML_CRIT_EXISTS

The requested criterion does already exist. A criterion has been installed at this position.

SML_ALT_UNKNOWN

The alternative does not exist.

SML_ALT_MISMATCH

The added criterion does not have the same number of alternatives as the frame.

SML_NAME_MISSING

The frame has not been given a name pointer.

SML_NAME_TOO_LONG

The frame name has too many characters.

SML_NAME_EXISTS

The frame name exists already in another frame.

SML_STMT_ERROR

Syntax error in the input statement.

SML_WRONG_METHOD

The method field contains an illegal value.

SML_WRONG_TOLERANCE

The tolerance in the call is not within range.

SML_CRIT_MISSING

A criterion is missing in a PM-frame and stand-in evaluation is not allowed.

SML_TOO_FEW_ALTS

Too few alternatives were specified in the call.

SML_INCONSISTENT

The supplied statement is inconsistent.

SML_NOT_ALLOWED

The call is not allowed at this time.

SML_FILE_UNKNOWN

The supplied filename is not a file in the current folder.

SML_WEAK_MASS_DISTR

Due to skew in the belief mass, the distributions are compressed.

SML_USER_ABORT

The call was prematurely aborted by the user. No call results are available.

SML_BUSY

Two threads have called SML in parallel. Since the code is not re-entrant, his thread has to wait for the first to finish. Guard against mix-up of threads in the calling application.

SML_LOGFILE_ERROR

Unable to open or write to the call sequence trace log file.

SML_MEMORY_LEAK

At reconciliation time, allocated memory still remains in use even though it should all be freed. Internal error in SML.

SML_BUFFER_OVERRUN

The string supplied was too short for the data returned.

SML error numbers

SML_KERNEL_ERROR	-100
SML_INPUT_ERROR	-101
SML_TREE_ERROR	-102
SML_OUTPUT_ERROR	-103
SML_FRAME_EXISTS	-104
SML_FRAME_UNKNOWN	-105
SML_FRAME_IN_USE	-106
SML_FRAME_NOT_LOADED	-107
SML_FRAME_CORRUPT	-108
SML_WRONG_FRAME_TYPE	-109
SML_WRONG_STATEMENT_TYPE	-110
SML_CONS_OVERFLOW	-111
SML_CRIT_OVERFLOW	-112
SML_LOGFILE_ERROR	-113
SML_INCONSISTENT	-114
SML_DIFFERING_RANKS	-115
SML_STMT_ERROR	-116
SML_SYS_CORRUPT	-117
SML_ALT_OVERFLOW	-118
SML_NODE_OVERFLOW	-119
SML_CRIT_MISSING	-120
SML_TOO_FEW_ALTS	-121
SML_USER_ABORT	-122
SML_STATE_ERROR	-123
SML_CRIT_UNKNOWN	-124
SML_CRIT_EXISTS	-125
SML_ALT_UNKNOWN	-126
SML_ALT_MISMATCH	-127
SML_BUSY	-128
SML_NAME_MISSING	-129
SML_NAME_TOO_LONG	-130
SML_NAME_EXISTS	-131
SML_NOT_ALLOWED	-132
SML_WRONG_METHOD	-133
SML_WRONG_TOLERANCE	-134
SML_FILE_UNKNOWN	-135
SML_INTERNAL_ERROR	-137
SML_WEAK_MASS_DISTR	-138
SML_MEMORY_LEAK	-139
SML_BUFFER_OVERRUN	-140
SML_ASSERT_FAILED	-141

TCL error codes

In the event of a SML_KERNEL_ERROR, a problem with the request has been detected in the underlying TCL calculation kernel. TCL reports the error to

SML as a positive number not to interfere with SML error numbers. SML records the error and it is passed on to the SML caller as one numerical component in SML_KERNEL_ERROR. The possible codes are:

TCL_INCONSISTENT

The call results in a previously consistent frame becoming inconsistent. The call has been rolled back, and the frame is in the same state as it was before the call.

TCL_INPUT_ERROR

An input parameter contains illegal data, for example, an index out of range or values not within given intervals.

TCL_TREE_ERROR

The structure of the specified input tree is not a valid tree according to the syntactic requirements.

TCL_ILLEGAL_NODE

An attempt to assign a value to an intermediate node in a tree.
(Probabilities and weights are allowed but not values)

TCL_TOO_FEW_ALTS

The call contains too few alternatives. This should be prohibited in the user interface at an earlier point.

TCL_TOO_MANY_ALTS

The call contains too many alternatives. This should be prohibited in the user interface at an earlier point.

TCL_TOO_MANY_CONS

The call contains too many consequences. This should be prohibited in the user interface at an earlier point.

TCL_TOO_MANY_STMTS

The call contains too many statements. This should be prohibited in the user interface at an earlier point.

TCL_TOO_NARROW_STMT

The TCL layer could operate in a mode where, for reasons of speed and stability, intervals of very small size are not allowed. This excludes the use of pointwise statements.

TCL_ATTACHED

An attempt to delete a frame that is currently attached (loaded).

TCL_DETACHED

An attempt to access a frame that is currently detached (unloaded).

TCL_CORRUPTED

The frame or other system resources have been rendered corrupt, either by modifications outside of TCL or because of an internal error in TCL.

TCL_OUT_OF_MEMORY

The kernel has run out of memory. This is the result of allocating too little virtual memory to the application in which the TCL layer is hosted.

TCL_MEMORY_LEAK

Memory not recycled at garbage collection.

TCL error numbers

```
TCL_INCONSISTENT    1
TCL_INPUT_ERROR     2
TCL_TREE_ERROR      3
TCL_ILLEGAL_NODE    4
TCL_TOO_MANY_CONS   5
TCL_TOO_MANY_ALTS   6
TCL_TOO_MANY_STMTS  7
TCL_TOO_NARROW_STMT 8
TCL_TOO_FEW_ALTS    9
TCL_CORRUPTED      10
TCL_ATTACHED       11
TCL_DETACHED       12
TCL_OUT_OF_MEMORY  13
TCL_MEMORY_LEAK    14
```

Mapping of SML return codes

This is the mapping of SML return codes to the error interpretation done by SML_error2 and thus indirectly by all error checks above.

SML return codes	**Interpretation**	**SML_error2 value***
SML_OK	Output result valid	0
SML_DIFFERING_RANKS		
SML_WEAK_MASS_DISTR		
SML_USER_ABORT	Output result invalid	1
All other return codes	Error occurred	2
TCL return codes		
TCL_TOO_MANY_STMTS	Output result invalid	1
TCL_TOO_MANY_CONS		
All other return codes	Error occurred	2

***** NOTE: Only when the result value is 0 there exists a result from the call. Thus, only after an evaluation call resulting in the value 0 is the result cache filled and subsequent output calls such as belief mass will succeed.

Call sequence trace (log file)

SML contains the ability to create a log file (the call sequence trace log, cst_log). This log file contains all the API calls to SML and enables the possibility to trace how an application works from the outside. It can be configured to log only the calls or alternatively also the results of the calls. It is enabled by storing a file "call_seq.log" in the home directory of the application calling SML. The first line of text in the file controls the trace level and is shown in parenthesis below. Running under MS Windows, the text must be encoded in ANSI (not UTF-8).

```
Level 0 (no file or no text): no log file written
Level 1 ("call_seq.log"):     input data + execution status
Level 2 ("call_seq_ext.log"): level 1 + output data
```

For level 2, replacing the first line with "call_seq_exx.log" also turns the error trace on. Similarly, "call_seq_exy.log" turns the error trace on but not the call sequence trace.

API function acronyms

All API functions that alter the contents in SML or ask for an evaluation of the contents have an acronym that will show up in the cst_log file (if it is enabled) in case of runtime error or single thread violation, or in the system trace file (if cst_log is not enabled).

	System functions
INIT	SML_init
EXIT	SML_exit
	File functions
FREAD	SML_read_frame
FRDDT	SML_read_ddt_frame
FWRT	SML_write_frame
	Frame functions
PMF	SML_new_DM_flat_frame
PMT	SML_new_DM_tree_frame
PMT	SML_new_SM_tree_frame
PMF	SML_new_PM_flat_frame
PMT	SML_new_PM_tree_frame
PMCT	SML_new_PM_crit_tree
DPMC	SML_delete_PM_crit
DISP	SML_dispose_frame
LOAD	SML_load_frame
UNL	SML_unload_frame
	Weight functions
SWB	SML_set_W_base
GWH	SML_get_W_hull
	Probability functions
SPB	SML_set_P_base
GPH	SML_get_P_hull
	Value functions
SVM	SML_set_V_base
GVH	SML_get_V_hull
	Evaluation functions
EVAL	SML_evaluate_frame

OMEGA	SML_evaluate_omega
OMEGA1	SML_evaluate_omega1
COMP	SML_compare_alternatives
DMASS	SML_delta_mass
RANK	SML_rank_alternatives
DAISY	SML_daisy_chain
DAISY	SML_pie_chart
TOW	SML_get_W_tornado
TOP	SML_get_P_tornado
TMCP	SML_get_MCP_tornado
TOV	SML_get_V_tornado
TMCV	SML_get_MCV_tornado
BTP	SML_get_BTP_tornado
BTV	SML_get_BTV_tornado
CINF	SML_get_cons_influence
	Belief mass functions
AMASS	SML_get_mass_above
BMASS	SML_get_mass_below
RMASS	SML_get_mass_range
SMASS	SML_get_support_mass
SMASL	SML_get_support_lower
SMASU	SML_get_support_upper