



Informatica® Identity Resolution
10.1

Developer Guide

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2018-06-28

Contents

Table of Contents	3
Introduction	18
Process Overview	19
Concepts	19
Connections	19
Sessions	19
Systems	19
Searches	19
Basic API	19
Advanced API	20
Sample API Programs	21
Sample System	21
Directory Structure	21
Population	21
Installing	21
IDT Layout	22
Building the Programs	22
Language specific notes	22
Running the Samples	22
C	23
C#	23
Java	23
Perl	23
VB.NET	23
Sample 1 - Basic API	24
Logic	24
Open a socket to the <i>Search</i> Server	24
Open a System and start a Search	24
Retrieve Result Set	24
Terminate the Search and close the System	24
Disconnect from the Search Server	25
Sample 2 - Advanced API	25
Logic	25
Open a socket to the <i>Connection</i> Server	25
Open a Session	25
Open a System	25
Retrieve the Input-View layout	26
for each Search	26
Close System	26
Close Session (optional)	27
Disconnect from the Connection Server	27
API Reference	28
Data Types	28
Strings	28
String Arrays	28
Blocks	29
Block Arrays	29
Nulls and NULs	29
Error Handling	29

Calling from C	30
Constants	30
Parameter types	30
ids_addr_get_cass_field	30
ids_addr_get_cass_field_cnt	31
ids_addr_get_cass_field_info	31
ids_addr_get_del_lines_ext	32
ids_addr_get_field	32
ids_addr_get_field_count	33
ids_addr_get_field_ext	34
ids_addr_get_field_idx	34
ids_addr_get_field_info_ext	35
ids_addr_get_field_len	36
ids_addr_get_line_len	36
ids_addr_get_option	37
ids_addr_info	37
ids_addr_init	38
ids_addr_parse	38
ids_addr_preload_country	39
ids_addr_set_attrib	39
ids_addr_set_del_lines	40
ids_addr_set_field_case	41
ids_addr_set_field_ext	41
ids_addr_set_field_idx	42
ids_addr_set_field_name	42
ids_addr_set_lines	43
ids_addr_set_option	44
ids_addr_std	44
ids_addr_validate	45
ids_connect	46
ids_disconnect	46
ids_error_get	47
ids_errors_get_all	47
ids_identify	48
ids_is_little_endian	48
ids_match_explain	49
ids_match_explain_count	49
ids_real_time_async_get	50
ids_real_time_async_start	51
ids_real_time_flul_add	52
ids_real_time_flul_close	53
ids_real_time_flul_delete	53
ids_real_time_flul_find_rule	54
ids_real_time_flul_get_rule	54
ids_real_time_flul_init	55
ids_real_time_sync_get	55
ids_real_time_sync_start	56
ids_scores_get	57
ids_search_comment_get	58
ids_search_dedupe_start	58
ids_search_fields_count	59
ids_search_fields_get	60
ids_search_filter	60
ids_search_finish	61
ids_search_get	61

ids_search_get_complete	62
ids_search_get_detail	63
ids_search_IDT_get	64
ids_search_layout	64
ids_search_profile_count	66
ids_search_profile_get	66
ids_search_start	67
ids_search_start_via_parameters	68
ids_search_start_via_record	69
ids_search_tolerances_count	70
ids_search_tolerances_get	70
ids_search_view_get	71
ids_search_view_set	72
ids_search_widths_count	72
ids_search_widths_get	73
ids_server_version_get	73
ids_session_close	74
ids_session_open	74
ids_set_encoding	75
ids_set_timeout	75
ids_set_vpd_user	76
ids_system_close	76
ids_system_idtname_count	77
ids_system_idtname_get	77
ids_system_notify	78
ids_system_open	78
ids_system_search_finish	79
ids_system_search_start	79
ids_system_searches_count	80
ids_system_searches_get	81
ids_systems_count	81
ids_systems_get	82

Calling from C without Arrays 83

Constants	83
Parameter types	83
ids_addr_get_cass_field	83
ids_addr_get_cass_field_cnt	84
ids_addr_get_cass_field_info	84
ids_addr_get_del_lines_ext	85
ids_addr_get_field	86
ids_addr_get_field_count	86
ids_addr_get_field_ext	87
ids_addr_get_field_idx	87
ids_addr_get_field_info_ext	88
ids_addr_get_field_len	89
ids_addr_get_line_len	89
ids_addr_get_option	90
ids_addr_info	90
ids_addr_init	91
ids_addr_parse	91
ids_addr_preload_country	92
ids_addr_set_attrib	92
ids_addr_set_del_lines	93
ids_addr_set_field_case	93
ids_addr_set_field_ext	94

ids_addr_set_field_idx	95
ids_addr_set_field_name	95
ids_addr_set_lines	96
ids_addr_set_option	97
ids_addr_std	97
ids_addr_validate	98
ids_connect	98
ids_disconnect	99
ids_error_get	99
ids_errors_get_all	100
ids_identify	100
ids_is_little_endian	101
ids_match_explain	101
ids_match_explain_count	102
ids_real_time_async_get	103
ids_real_time_async_start	103
ids_real_time_flul_add	104
ids_real_time_flul_close	105
ids_real_time_flul_delete	106
ids_real_time_flul_find_rule	106
ids_real_time_flul_get_rule	107
ids_real_time_flul_init	107
ids_real_time_sync_get	108
ids_real_time_sync_start	108
ids_scores_get	110
ids_search_comment_get	110
ids_search_dedupe_start	111
ids_search_fields_count	112
ids_search_fields_get	112
ids_search_filter	113
ids_search_finish	113
ids_search_get	114
ids_search_get_complete	114
ids_search_get_detail	115
ids_search_IDT_get	116
ids_search_layout	117
ids_search_profile_count	118
ids_search_profile_get	118
ids_search_start	119
ids_search_start_via_parameters	120
ids_search_start_via_record	121
ids_search_tolerances_count	122
ids_search_tolerances_get	122
ids_search_view_get	123
ids_search_view_set	123
ids_search_widths_count	124
ids_search_widths_get	124
ids_server_version_get	125
ids_session_close	125
ids_session_open	126
ids_set_encoding	126
ids_set_timeout	127
ids_set_vpd_user	127
ids_system_close	128
ids_system_idtname_count	128

ids_system_idtname_get	129
ids_system_notify	129
ids_system_open	130
ids_system_search_finish	130
ids_system_search_start	131
ids_system_searches_count	132
ids_system_searches_get	132
ids_systems_count	133
ids_systems_get	133

Calling from C# 135

Installation - Win32 client	135
Constants	135
Response code	135
Parameter types	135
Constructor	135
addr_get_cass_field	136
addr_get_cass_field_cnt	136
addr_get_cass_field_info	137
addr_get_del_lines_ext	137
addr_get_field	138
addr_get_field_count	139
addr_get_field_ext	139
addr_get_field_idx	140
addr_get_field_info_ext	140
addr_get_field_len	141
addr_get_line_len	141
addr_get_option	142
addr_info	142
addr_init	143
addr_parse	143
addr_preload_country	144
addr_set_attrib	144
addr_set_del_lines	145
addr_set_field_case	146
addr_set_field_ext	146
addr_set_field_idx	147
addr_set_field_name	147
addr_set_lines	148
addr_set_option	149
addr_std	149
addr_validate	150
disconnect	150
error_get	151
errors_get_all	151
identify	152
is_little_endian	152
match_explain	153
match_explain_count	153
real_time_async_get	154
real_time_async_start	154
real_time_flul_add	156
real_time_flul_close	156
real_time_flul_delete	157
real_time_flul_find_rule	157
real_time_flul_get_rule	158

real_time_flul_init	158
real_time_sync_get	159
real_time_sync_start	159
scores_get	161
search_comment_get	161
search_dedupe_start	162
search_fields_count	163
search_fields_get	163
search_filter	164
search_finish	164
search_get	165
search_get_complete	165
search_get_detail	166
search_IDT_get	167
search_layout	167
search_profile_count	168
search_profile_get	169
search_start	169
search_start_via_parameters	170
search_start_via_record	171
search_tolerances_count	172
search_tolerances_get	172
search_view_get	173
search_view_set	174
search_widths_count	174
search_widths_get	175
server_version_get	175
session_close	175
session_open	176
set_encoding	176
set_timeout	177
set_vpd_user	177
system_close	178
system_idtname_count	178
system_idtname_get	179
system_notify	179
system_open	180
system_search_finish	180
system_search_start	181
system_searches_count	182
system_searches_get	182
systems_count	183
systems_get	183

Calling from Cobol (z/OS)	184
Constants	184
Installation	184
Installation (TSO/batch)	184
Installation (CICS) -static calls	185
Installation (CICS) - dynamic calls	185
Defining programs to CICS	185
Parameter types	185
IDS-ADDR-GET-CASS-FIELD	186
IDS-ADDR-GET-CASS-FIELD-CNT	186
IDS-ADDR-GET-CASS-FIELD-INFO	187
IDS-ADDR-GET-DEL-LINES-EXT	187

IDS-ADDR-GET-FIELD	188
IDS-ADDR-GET-FIELD-COUNT	189
IDS-ADDR-GET-FIELD-EXT	189
IDS-ADDR-GET-FIELD-IDX	190
IDS-ADDR-GET-FIELD-INFO-EXT	191
IDS-ADDR-GET-FIELD-LEN	191
IDS-ADDR-GET-LINE-LEN	192
IDS-ADDR-GET-OPTION	192
IDS-ADDR-INFO	193
IDS-ADDR-INIT	193
IDS-ADDR-PARSE	194
IDS-ADDR-PRELOAD-COUNTRY	194
IDS-ADDR-SET-ATTRIB	195
IDS-ADDR-SET-DEL-LINES	196
IDS-ADDR-SET-FIELD-CASE	196
IDS-ADDR-SET-FIELD-EXT	197
IDS-ADDR-SET-FIELD-IDX	197
IDS-ADDR-SET-FIELD-NAME	198
IDS-ADDR-SET-LINES	198
IDS-ADDR-SET-OPTION	199
IDS-ADDR-STD	200
IDS-ADDR-VALIDATE	201
IDS-CONNECT	201
IDS-DISCONNECT	202
IDS-ERROR-GET	202
IDS-ERRORS-GET-ALL	203
IDS-IDENTIFY	203
IDS-IS-LITTLE-ENDIAN	204
IDS-MATCH-EXPLAIN	204
IDS-MATCH-EXPLAIN-COUNT	205
IDS-REAL-TIME-ASYNC-GET	205
IDS-REAL-TIME-ASYNC-START	206
IDS-REAL-TIME-FLUL-ADD	207
IDS-REAL-TIME-FLUL-CLOSE	208
IDS-REAL-TIME-FLUL-DELETE	208
IDS-REAL-TIME-FLUL-FIND-RULE	209
IDS-REAL-TIME-FLUL-GET-RULE	209
IDS-REAL-TIME-FLUL-INIT	210
IDS-REAL-TIME-SYNC-GET	210
IDS-REAL-TIME-SYNC-START	211
IDS-SCORES-GET	212
IDS-SEARCH-COMMENT-GET	213
IDS-SEARCH-DEDUPE-START	213
IDS-SEARCH-FIELDS-COUNT	214
IDS-SEARCH-FIELDS-GET	215
IDS-SEARCH-FILTER	215
IDS-SEARCH-FINISH	216
IDS-SEARCH-GET	216
IDS-SEARCH-GET-COMPLETE	217
IDS-SEARCH-GET-DETAIL	218
IDS-SEARCH-IDT-GET	219
IDS-SEARCH-LAYOUT	219
IDS-SEARCH-PROFILE-COUNT	221
IDS-SEARCH-PROFILE-GET	221
IDS-SEARCH-START	222

IDS-SEARCH-START-VIA-PARAMETERS	223
IDS-SEARCH-START-VIA-RECORD	224
IDS-SEARCH-TOLERANCES-COUNT	225
IDS-SEARCH-TOLERANCES-GET	225
IDS-SEARCH-VIEW-GET	226
IDS-SEARCH-VIEW-SET	226
IDS-SEARCH-WIDTHS-COUNT	227
IDS-SEARCH-WIDTHS-GET	227
IDS-SERVER-VERSION-GET	228
IDS-SESSION-CLOSE	228
IDS-SESSION-OPEN	229
IDS-SET-ENCODING	229
IDS-SET-TIMEOUT	230
IDS-SET-VPD-USER	230
IDS-SYSTEM-CLOSE	231
IDS-SYSTEM-IDTNAME-COUNT	231
IDS-SYSTEM-IDTNAME-GET	232
IDS-SYSTEM-NOTIFY	232
IDS-SYSTEM-OPEN	233
IDS-SYSTEM-SEARCH-FINISH	234
IDS-SYSTEM-SEARCH-START	234
IDS-SYSTEM-SEARCHES-COUNT	235
IDS-SYSTEM-SEARCHES-GET	235
IDS-SYSTEMS-COUNT	236
IDS-SYSTEMS-GET	236
Calling from Java	238
Java Version	238
Overview	238
Constants	238
Parameter types	238
Error Handling	238
Deprecated APIs	240
ClieSock Constructor	240
ids_addr_get_cass_field	241
ids_addr_get_cass_field_cnt	242
ids_addr_get_cass_field_info	242
ids_addr_get_del_lines_ext	243
ids_addr_get_field	244
ids_addr_get_field_count	244
ids_addr_get_field_ext	245
ids_addr_get_field_idx	245
ids_addr_get_field_info_ext	246
ids_addr_get_field_len	247
ids_addr_get_line_len	247
ids_addr_get_option	248
ids_addr_info	248
ids_addr_init	249
ids_addr_parse	249
ids_addr_preload_country	250
ids_addr_set_attrib	250
ids_addr_set_del_lines	251
ids_addr_set_field_case	251
ids_addr_set_field_ext	252
ids_addr_set_field_idx	253
ids_addr_set_field_name	253

ids_addr_set_lines	254
ids_addr_set_option	254
ids_addr_std	255
ids_addr_validate	256
ids_disconnect	256
ids_error_get	257
ids_errors_get_all	257
ids_identify	258
ids_is_little_endian	258
ids_match_explain	259
ids_match_explain_count	260
ids_real_time_async_get	260
ids_real_time_async_start	261
ids_real_time_flul_add	262
ids_real_time_flul_close	263
ids_real_time_flul_delete	263
ids_real_time_flul_find_rule	264
ids_real_time_flul_get_rule	264
ids_real_time_flul_init	265
ids_real_time_sync_get	265
ids_real_time_sync_start	266
ids_scores_get	267
ids_search_comment_get	268
ids_search_dedupe_start	268
ids_search_fields_count	269
ids_search_fields_get	270
ids_search_filter	270
ids_search_finish	271
ids_search_get	271
ids_search_get_complete	272
ids_search_get_detail	273
ids_search_IDT_get	273
ids_search_layout	274
ids_search_profile_count	275
ids_search_profile_get	276
ids_search_start	276
ids_search_start_via_parameters	278
ids_search_start_via_record	278
ids_search_tolerances_count	279
ids_search_tolerances_get	280
ids_search_view_get	280
ids_search_view_set	281
ids_search_widths_count	282
ids_search_widths_get	282
ids_server_version_get	283
ids_session_close	283
ids_session_open	283
ids_set_encoding	284
ids_set_timeout	285
ids_set_vpd_user	285
ids_system_close	286
ids_system_idtname_count	286
ids_system_idtname_get	286
ids_system_notify	287
ids_system_open	287

ids_system_search_finish	288
ids_system_search_start	289
ids_system_searches_count	289
ids_system_searches_get	290
ids_systems_count	290
ids_systems_get	291

Calling from Perl 292

Constants - Object Oriented	292
Installation - Win32 client	292
Installation - Unix client	292
Parameter types	292
addr_get_cass_field	292
addr_get_cass_field_cnt	293
addr_get_cass_field_info	293
addr_get_del_lines_ext	294
addr_get_field	295
addr_get_field_count	295
addr_get_field_ext	296
addr_get_field_idx	296
addr_get_field_info_ext	297
addr_get_field_len	298
addr_get_line_len	298
addr_get_option	298
addr_info	299
addr_init	299
addr_parse	300
addr_preload_country	300
addr_set_attrb	301
addr_set_del_lines	301
addr_set_field_case	302
addr_set_field_ext	303
addr_set_field_idx	303
addr_set_field_name	304
addr_set_lines	304
addr_set_option	305
addr_std	306
addr_validate	307
disconnect	307
error_get	308
errors_get_all	308
identify	309
is_little_endian	309
match_explain	310
match_explain_count	310
real_time_async_get	311
real_time_async_start	311
real_time_flul_add	313
real_time_flul_close	313
real_time_flul_delete	314
real_time_flul_find_rule	314
real_time_flul_get_rule	315
real_time_flul_init	315
real_time_sync_get	316
real_time_sync_start	316
scores_get	318

search_comment_get	318
search_dedupe_start	319
search_fields_count	320
search_fields_get	320
search_filter	321
search_finish	321
search_get	322
search_get_complete	322
search_get_detail	323
search_IDT_get	324
search_layout	324
search_profile_count	325
search_profile_get	326
search_start	326
search_start_via_parameters	327
search_start_via_record	328
search_tolerances_count	329
search_tolerances_get	329
search_view_get	330
search_view_set	331
search_widths_count	331
search_widths_get	332
server_version_get	332
session_close	332
session_open	333
set_encoding	333
set_timeout	334
set_vpd_user	334
system_close	335
system_idtname_count	335
system_idtname_get	336
system_notify	336
system_open	337
system_search_finish	337
system_search_start	338
system_searches_count	339
system_searches_get	339
systems_count	340
systems_get	340

Calling from Visual Basic .NET	341
Constants	341
Installation - Win32 client	341
Constructor	342
addr_get_cass_field	342
addr_get_cass_field_cnt	343
addr_get_cass_field_info	343
addr_get_del_lines_ext	344
addr_get_field	344
addr_get_field_count	345
addr_get_field_ext	345
addr_get_field_idx	346
addr_get_field_info_ext	347
addr_get_field_len	347
addr_get_line_len	348
addr_get_option	348

addr_info	349
addr_init	349
addr_parse	350
addr_preload_country	350
addr_set_attrb	351
addr_set_del_lines	351
addr_set_field_case	352
addr_set_field_ext	352
addr_set_field_idx	353
addr_set_field_name	354
addr_set_lines	354
addr_set_option	355
addr_std	355
addr_validate	356
disconnect	357
error_get	357
errors_get_all	358
identify	358
is_little_endian	359
match_explain	359
match_explain_count	360
real_time_async_get	360
real_time_async_start	361
real_time_flul_add	362
real_time_flul_close	363
real_time_flul_delete	363
real_time_flul_find_rule	364
real_time_flul_get_rule	364
real_time_flul_init	365
real_time_sync_get	365
real_time_sync_start	366
scores_get	367
search_comment_get	367
search_dedupe_start	368
search_fields_count	369
search_fields_get	369
search_filter	370
search_finish	370
search_get	371
search_get_complete	372
search_get_detail	372
search_IDT_get	373
search_layout	373
search_profile_count	374
search_profile_get	375
search_start	375
search_start_via_parameters	377
search_start_via_record	377
search_tolerances_count	378
search_tolerances_get	379
search_view_get	379
search_view_set	380
search_widths_count	380
search_widths_get	381
server_version_get	381

session_close	382
session_open	382
set_encoding	383
set_timeout	383
set_vpd_user	384
system_close	384
system_idtname_count	385
system_idtname_get	385
system_notify	385
system_open	386
system_search_finish	387
system_search_start	387
system_searches_count	388
system_searches_get	388
systems_count	389
systems_get	389

Using IIR with XML 391

WSDL	391
Creating a proxy with .NET	391
Creating a proxy with Apache Axis2	391
XML and HTTP	391
XML and SOAP	392
XML and Unicode	394
Parameter types	394
ids_addr_get_cass_field	394
ids_addr_get_cass_field_cnt	395
ids_addr_get_cass_field_info	396
ids_addr_get_del_lines_ext	397
ids_addr_get_field	398
ids_addr_get_field_count	399
ids_addr_get_field_ext	400
ids_addr_get_field_idx	401
ids_addr_get_field_info_ext	402
ids_addr_get_field_len	403
ids_addr_get_line_len	404
ids_addr_get_option	405
ids_addr_info	406
ids_addr_init	407
ids_addr_parse	408
ids_addr_preload_country	408
ids_addr_set_attrib	409
ids_addr_set_del_lines	410
ids_addr_set_field_case	411
ids_addr_set_field_ext	412
ids_addr_set_field_idx	413
ids_addr_set_field_name	414
ids_addr_set_lines	415
ids_addr_set_option	417
ids_addr_std	417
ids_addr_validate	419
ids_connect	420
ids_disconnect	420
ids_error_get	421
ids_errors_get_all	422
ids_identify	423

ids_is_little_endian	424
ids_match_explain	425
ids_match_explain_count	426
ids_real_time_async_get	427
ids_real_time_async_start	428
ids_real_time_flul_add	429
ids_real_time_flul_close	430
ids_real_time_flul_delete	431
ids_real_time_flul_find_rule	432
ids_real_time_flul_get_rule	433
ids_real_time_flul_init	434
ids_real_time_sync_get	435
ids_real_time_sync_start	436
ids_scores_get	438
ids_search_comment_get	439
ids_search_dedupe_start	440
ids_search_fields_count	441
ids_search_fields_get	442
ids_search_filter	443
ids_search_finish	444
ids_search_get	445
ids_search_get_complete	446
ids_search_get_detail	447
ids_search_IDT_get	448
ids_search_layout	449
ids_search_profile_count	451
ids_search_profile_get	452
ids_search_start	453
ids_search_start_via_parameters	455
ids_search_start_via_record	456
ids_search_tolerances_count	457
ids_search_tolerances_get	458
ids_search_view_get	459
ids_search_view_set	460
ids_search_widths_count	461
ids_search_widths_get	462
ids_server_version_get	463
ids_session_close	464
ids_session_open	464
ids_set_encoding	465
ids_set_timeout	466
ids_set_vpd_user	467
ids_system_close	468
ids_system_idtname_count	469
ids_system_idtname_get	470
ids_system_notify	471
ids_system_open	472
ids_system_search_finish	473
ids_system_search_start	473
ids_system_searches_count	475
ids_system_searches_get	476
ids_systems_count	476
ids_systems_get	477
Common Parameters	479
Verbosity	479

Address Standardization	480
Initialization	480
Character Sets and Countries	480
Providing an Input Address	486
Parsing an Address	487
Validating an Address	488
Retrieving Address Fields	490
Setting Options	492
Getting Options	493
Sample Code	493
Validation Database Files	493
ASM configuration for AddressDoctor v5	494
Batch Test Utility	496
AnswerSet	499
Table Name and Index	499
Table Layout	499
Adding Rows	499
Clearing the Table	500
Relate & DupFinder Set-Ids	500
Oracle RAC	500
DupFinder	501
AnswerSet	502
Match Explain API	503
Match Explain Record Formats	503
Match Explain Scores	504
Index	505

Introduction

This manual describes how to develop a custom search client application using the API.

Prototype search clients are provided in the following languages:

- ♦ C
- ♦ C#
- ♦ Cobol
- ♦ Java
- ♦ Perl
- ♦ Visual Basic .NET

Process Overview

Concepts

Connections

The `ids_connect` and `ids_disconnect` API calls are used to establish / terminate a connection (TCP/IP socket) to the "server". The "server" can be either the Search Server or Connection Server, depending on the host name and host port parameters supplied on the connection call.

Sessions

Use of the Connection Server is optional. Refer to the OPERATIONS manual, *Servers* chapter for details about the Connection Server and why you might want to use it.

If it is used, you must establish a "session" with the Connection Server using the `ids_session_open` API.

Sessions may be closed explicitly using the `ids_session_close` API, or can be timed-out by the Connection Server if they have been inactive for the defined time-out period.

Systems

The `ids_system_open` and `ids_system_close` calls are used to inform the server which System will be used for subsequent search calls. Switching between different Systems incurs a slight overhead, as the Rulebase must be consulted to read the rules for the new System.

Searches

Searches are started with `ids_search_start`. This call performs all of the work necessary to find candidates and match them against the search record. The resulting set of records is created and sorted. The call returns the number of records in the resulting set.

The results may be written to a database table known as an AnswerSet , or may be retrieved using `ids_search_get`. The latter API is called repeatedly to retrieve records from the result set one at a time.

Once enough rows have been retrieved, any remaining resources held by the server for this search are freed using `ids_search_finish`.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be architected to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Basic API

The Basic API is suitable for processing a simple one-off query. It combines opening the System with starting a search. It also combines terminating a search and closing a System in one call.

Use the Advanced API if you wish to start multiple searches for the same System.

The flow of the Basic API is

1. Open a connection (`ids_connect`)
2. Initiate a search (`ids_system_search_start`)
3. Retrieve sorted search results (`ids_search_get`) until required buffer is filled (this may be a screen's worth, or a pre-determined search limit size).
4. Process the search results (display to a screen or write to a file)
5. End the search (`ids_system_search_finish`)
6. Close the connection (`ids_disconnect`)

Advanced API

The Advanced API is more efficient than the Basic API but requires the use of two extra calls. It is suited to applications that wish to perform multiple searches for a given System.

The flow of the more advanced level is

1. Open a Socket (`ids_connect`)
2. Optionally start a Session (`ids_session_open`)
3. Open a System (`ids_system_open`)
4. Initiate a Search (`ids_search_start`)
5. Retrieve sorted search results (`ids_search_get`) until required buffer is filled (this may be a screen's worth, or a pre-determined search limit size).
6. Process the search results (display to a screen or write to a file)
7. End the search (`ids_search_finish`)
8. Optionally return to 4. to start more searches
9. Close the system (`ids_system_close`)
10. Optionally close the session (`ids_session_close`)
11. Close the socket (`ids_disconnect`)

Other functions are available through this level of the API allowing the search client to be more rule-sensitive. For example, the length of a record to be returned in the search results can be determined with an `ids_search_view_get` call which is followed by a call to `ids_search_layout`. This may assist the client's dynamic memory allocation.

Sample API Programs

Sample client programs are provided for all supported API languages. All sample programs use the same System (and data) and demonstrate the same logical sequence of API calls.

Sample System

All sample programs use the same System (`ssa001`). This is provided as an SDF.

Directory Structure

The sample System, programs and data are located in the **samples** directory of the Client and Developer Components installation directory. The following directory structure exists:

```
samples/  
  data/  
    f5000.txt  
  programs/  
    applet/  
    c/  
    cobol/  
    csharp/  
    java/  
    perl/  
    perl-oo/  
    vbnet/  
  system/  
    ssa001.sdf
```

Population

The sample System uses the same population as the IIR Installation test. The SSA-NAME3 System is named `iirXXXX`, where `XXXX` is the release number of IIR. The Population is named `test`.

Installing

- ◆ Launch IIR Console and choose 'New' from the 'System' menu.
- ◆ Choose 'Create a system from an SDF' and click 'OK'.
- ◆ Enter 'ssa001' as the system name, and `%SSATOP%\samples\system\ssa001.sdf` as the System Definition File (where `%SSATOP%` is the directory that the Client and Developer Components were installed into - by default, `c:\InformaticaIR`).
- ◆ Fill out your database details and click 'OK'.
- ◆ Click 'Close' when the system has been created.
- ◆ Under the 'System' menu, choose 'Select', then 'ssa001' and click 'OK'.
- ◆ Under the 'System' menu, choose 'Load IDT', then 'ssa001' and click 'OK'.

Your IIR installation is now ready to run the samples.

IDT Layout

The IDT has the following layout. Records passed to `ids_search_start` must be preformatted by the caller in this layout.

Name	Offset	Length
ID	0	11
name	11	50
DOB	61	8
Address	69	40

IDT record length is 109 bytes.

Building the Programs

(Instructions and files are only provided for the WINDOWS environment. Unix users will need to adjust this section accordingly).

To build the sample applications you will need to establish the client environment by running *<IIR Client Install Dir>\env\issc.bat*.

Once your environment is established, simply run **compile.bat** in the subdirectory of the sample you wish to build. The batch file assumes that the compilers are in your `PATH`. You will need to adjust your `PATH` environment variable if they are not.

Note that Perl is either interpreted or compiled at runtime, so no **compile.bat** file exists for these.

Language specific notes

C Ensure that **cl.exe** can be found in your path, and that the `INCLUDE` and `LIB` environment variables are set correctly.

C# Ensure that '**csc.exe**' can be found in your `PATH`.

VB.NET Ensure that **vbc.exe** can be found in your `PATH`.

Running the Samples

The commands below illustrate how to run the samples.

The *<RULEBASE>* parameter defines which Rulebase to use and is formatted as *type:method:details*, eg. `odb:0:user/password@hostname`

The *<HOST>* field is formatted as *hostname:port*, eg. `localhost:1666`

Sample 1 expects the host parameter to point to the Search Server, while Sample 2 expects the host parameter to point to the Connection Server.

The *<WORKDIR>* parameter defines which working directory to use.

The *<SYSTEM>* parameter defines the name of the system in the rulebase.

C

```
sample1 -r<RULEBASE> -h<HOST>
sample2 -r<RULEBASE> -h<HOST>
```

Note: the C examples use the API stub DLL **ssasec.dll** and the IIR socket interface provided by **ssaiok.dll**. These dynamic link libraries are packaged in the IIR Client **bin** and SSA-NAME3 **bin** directory respectively. Make sure these directories are in your PATH before running the samples.

C#

Load the .NET assembly into the system cache:

```
gacutil /i %SSATOP%\bin\ssasecs.dll
```

You can now execute the programs as follows:

```
sample1 -r<RULEBASE> -h<HOST>
sample2 -r<RULEBASE> -h<HOST>
```

Java

Add %SSATOP%\bin\idsclie.jar;%SSATOP%\bin\idssecl.jar;. to your CLASSPATH environment variable. Then run

```
java Sample1 -r<RULEBASE> -h<HOST>
java Sample2 -r<RULEBASE> -h<HOST>
```

Perl

```
perl sample1 -r<RULEBASE> -h<HOST>
perl sample2 -r<RULEBASE> -h<HOST>
```

VB.NET

Load the .NET assembly into the system cache:

```
gacutil /i %SSATOP%\bin\ssasecs.dll
```

You can now execute the programs as follows:

```
sample1 -r<RULEBASE> -h<HOST>
sample2 -r<RULEBASE> -h<HOST>
```

Sample 1 - Basic API

Sample 1 demonstrates the use of the Basic API calls. It also demonstrates how to start a search with an array of *field* values.

Logic

Open a socket to the *Search Server*

- ◆ API call: `ids_connect`
- ◆ This opens a socket (communication channel) to the Search Server using TCP/IP.
- ◆ The Server's host name (or IP address) and port number (1666 by default) are specified as parameters to this call.
- ◆ The Search Server is used instead of the Connection Server because this sample does not make use of the sessions (refer the Sample 2 for an example of how to use sessions).
- ◆ The call returns a *socket handle* that must be used on subsequent API calls to communicate over the same channel.

Open a System and start a Search

- ◆ API call: `ids_system_search_start`
- ◆ This call opens a System in preparation for a single search.
- ◆ The name of a valid Search-Definition defined in the System is used to initialize a Search.
- ◆ The search fields used in this example are passed to IIR using a Block Array.
- ◆ IIR will build a set of file records that match the search record and sort them in descending order of score (by default). The API call will return a count of the number of records in the result set.

Retrieve Result Set

- ◆ API call: `ids_search_get`
- ◆ This API is called repeatedly to retrieve one file record at a time.
- ◆ Each record returned is in IDT format by default.
- ◆ Records can be returned using an output view by calling `ids_IDT_views_output_set` prior to starting the search.
- ◆ Response code 1 is returned when the End-Of-Set has been reached.
- ◆ It is not mandatory to retrieve all records from the result set (although it would be unusual not to).

Terminate the Search and close the System

- ◆ API call: `ids_system_search_finish`
- ◆ This releases resources held by the Search Server for the Search and System.

Disconnect from the Search Server

- ◆ API call: `ids_disconnect`
- ◆ Closes the socket connection to the Server.

Sample 2 - Advanced API

Sample 2 demonstrates the use of the Advanced API to make slightly more complex (but more efficient) API calls. It demonstrates the use of sessions and how to start a search with a pre-constructed search record.

Logic

Open a socket to the *Connection* Server

- ◆ API call: `ids_connect`
- ◆ This opens a socket (communication channel) to the Connection Server using TCP/IP.
- ◆ The host name (or IP address) and port number (1667 by default) of the Connection Server are specified as parameters to the call.
- ◆ The Connection Server should be running on the same machine as the client application program (for maximum performance).
- ◆ The Connection Server is used instead of the Search Server because this sample will make use of sessions; a facility implemented by the Connection Server.
- ◆ The call returns a *socket handle* that must be used on subsequent API calls to communicate over the same channel.

Open a Session

- ◆ API call: `ids_session_open`
- ◆ This API call is used to establish a new session with the Connection Server or to reopen an existing session (created by a previous call).
- ◆ Specify a session number of -1 to create a new session, or the existing session number to reopen a previously created session.
- ◆ Sessions are used to identify resources on the Search Server and to keep them open even when the client application disconnects from the Connection Server.
- ◆ Stateless Web based transactions can store/retrieve the session number using a cookie.

Open a System

- ◆ API call: `ids_system_open`
- ◆ This call retrieves the System's rules from the Rulebase and prepares for making multiple searches.
- ◆ This is a relatively expensive operation so it is beneficial to reduce the number of times a system is opened/closed and/or switched.

Retrieve the Input-View layout

- ◆ API call: `ids_search_view_get`
- ◆ This returns the name, number of fields and record length of the current input-view associated with this Search.
- ◆ This followed by a call to `ids_search_layout`
- ◆ It returns arrays describing the field names, lengths and offsets. As no input (or output) views have been explicitly set, both views use the default format, which is the IDT layout.
- ◆ This information is typically used to allocate appropriately sized buffers and/or prompt a client for search data.

for each Search

- ◆ Get input data for each field required for search

Start a Search

- ◆ API call: `ids_search_start`
- ◆ **An input record is constructed from the search data. It must match the layout of the input-view.**

Retrieve Result Set

- ◆ API call: `ids_search_get`
- ◆ This API is called repeatedly to retrieve one record at a time.
- ◆ Each record returned is in IDT format.
- ◆ Response code 1 is returned when the End-Of-Set has been reached.
- ◆ It is not mandatory to retrieve all records from the result set (although it would be unusual not to).

Terminate Search

- ◆ API call: `ids_system_search_finish`
- ◆ This releases resources held by the Search Server for the Search and System.

End-Loop

Close System

- ◆ API call: `ids_system_close`
- ◆ Frees resources held on Search Server.

Close Session (optional)

- ♦ API call: `ids_session_close`
- ♦ This call instructs the Connection Server to free resources held on the Search Server and to invalidate the session number.
- ♦ A stateless Web based search will not issue this call unless it will definitely not attempt to reconnect to the Connection Server using the same session number.
- ♦ **Sessions that have not been closed explicitly with this call will be timed-out (closed) if they remain inactive for the designated time-out period of the Connection Server.**

Disconnect from the Connection Server

- ♦ API call: `ids_disconnect`
- ♦ Closes the socket connection to the Server.

API Reference

The API allows user written application programs to access functions that provide name-searching facilities over a network.

The API functions are callable from C, C#, Java, Perl and other Application Languages capable of calling a DLL. It is also available as an ActiveX control for use in languages such as Visual Basic.

The API is thread safe with the following exception: each thread must establish its own connection to the Search Server. The behavior of the API with multiple threads accessing the same connection to the Search Server is undefined.

Data Types

The section discusses the IIR data types used in this document. The *Language Specific Bindings* section describes the mapping between the IIR data types and the native data types used for specific programming languages.

Strings

The `String` data-type is a variable length piece of memory, terminated with a NUL character (0x00).

When using a `String` as an input parameter there is no need to explicitly tell IIR how long it is because IIR can detect its length.

When the API returns a `String` as an output parameter, the caller must allocate memory for it and in some programming languages, tell IIR how long it is. For example, in C it is not possible to detect how long a piece of memory is, so the caller must pass two parameters:

- ♦ the address of the memory, and
- ♦ its length.

IIR uses this information to prevent overwriting unallocated memory (which would result in GPF or core dump).

String Arrays

A `StringArray` is an array of `Strings`. It consists of an array of pointers, with each pointer pointing to a `String`. A `String Array` is usually passed through the API using three parameters:

- ♦ address of the array of pointers,
- ♦ number of pointers in the array, and
- ♦ the length of each `String`.

Note that since there is provision for only one length value, all `Strings` must be the same length.

Blocks

A `Block` data-type is a fixed length piece of memory. It is not NUL terminated. If a value is not long enough to fill the entire Block, the Block should be padded with spaces on the right. Since IIR cannot detect how long the memory is, Blocks are usually passed through the API using two parameters:

- ♦ a pointer to the memory, and
- ♦ the length of the Block.

Block Arrays

A `BlockArray` is an array of Blocks. It consists of an array of pointers, with each pointer pointing to a Block. A Block Array is usually passed through the API using three parameters:

- ♦ address of the array of pointers,
- ♦ number of pointers in the array, and
- ♦ the length of each Block.

Note that since there is provision for only one length value, all Blocks must be the same length.

Nulls and NULs

Some programming languages permit the use of Null pointers (such as C). Null pointers must never be passed as arguments to any API functions. If you do not wish to provide a value for an argument, use a NUL terminated (0x00) string instead (a zero-length, empty string).

Error Handling

An API program that receives a negative response code from a call should retrieve the associated error messages from the Server and log them.

The API function `ids_errors_get_all` is called repeatedly to retrieve one message at a time. It returns a positive response code when there are no more messages left to retrieve.

Calling from C

For Win32, the C API functions are prototyped in `%SSAINC%\ssasec1.h`. An import library `%SSALIB%\stssasec.lib` is used to link to the application. At run time it will dynamically load `ssasec.dll` and `ssaio.k.dll` from the IIR Client `bin` and SSA-NAME3 `bin` directories respectively.

For Unix, the C API functions are prototyped in `SSAINC/ssasec1.h` and shared libraries are found in `SSABIN` (`libssasec.so` and `libssaio.k.so`).

The functions return a response code. A negative response code indicates a transport error, after which the communication channel is closed and no further API calls can be made without reconnecting.

Constants

Constants are declared as `#defines`, in uppercase using underscores. Constants are prefixed with 'SSA_', eg. `SSA_MSG_SIZE`.

Parameter types

Note: values in **BOLD** represent information that must be provided to the function.

ids_addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
long ids_addr_get_cass_field (
    long sockh,           // Long in
    long suggest_idx,     // Long in
    long field_idx,       // Long in
    char * field_value,   // Block out
    long field_value_size
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
long ids_addr_get_cass_field_cnt (
    long sockh,          // Long in
    long * count          // Long out
);
```

Parameters:

sockh is the socket to use for the call

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
long ids_addr_get_cass_field_info (
    long sockh,          // Long in
    long suggest_idx,    // Long in
    long * field_length, // LongArray out
    long field_length_num
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

ids_addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Prototype:

```
long ids_addr_get_del_lines_ext (
    long sockh,           // Long in
    long suggest_idx,     // Long in
    long del_case,        // Long in
    char * del_line1,     // Block out
    long del_line1_size,
    char * del_line2,     // Block out
    long del_line2_size,
    char * del_line3,     // Block out
    long del_line3_size,
    char * del_line4,     // Block out
    long del_line4_size,
    char * del_line5,     // Block out
    long del_line5_size,
    char * del_line6,     // Block out
    long del_line6_size
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

ids_addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned

by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```
long ids_addr_get_field (
    long sockh,          // Long in
    long suggest_idx,    // Long in
    long field_idx,      // Long in
    char * field_value,  // Block out
    long field_value_size,
    long * field_val_status, // Long out
    long * field_val_mods // Long out
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

ids_addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
long ids_addr_get_field_count (
    long sockh,          // Long in
    long * count         // Long out
);
```

Parameters:

sockh is the socket to use for the call

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

ids_addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Prototype:

```
long ids_addr_get_field_ext (
    long sockh,           // Long in
    long suggest_idx,     // Long in
    long field_operation, // Long in
    char * field_name,    // String in
    long field_item_line, // Long in
    char * field_type,    // String in
    char * field_value,   // Block out
    long field_value_size
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

ids_addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
long ids_addr_get_field_idx (
    long sockh,          // Long in
    long suggest_idx,    // Long in
    long field_idx,      // Long in
    char * field_value,  // Block out
    long field_value_size
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

ids_addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
long ids_addr_get_field_info_ext (
    long sockh,          // Long in
    long suggest_idx,    // Long in
    long * field_length, // LongArray out
    long field_length_num,
    char * addr_label_encoded, // Block out
    long addr_label_encoded_size,
    char * addr_label_charset, // String out
    long addr_label_charset_size,
    long * score           // Long out
);
```

Parameters:

sockh is the socket to use for the call

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

ids_addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

```
long ids_addr_get_field_len (
    long sockh,          // Long in
    long * max_len       // Long out
);
```

Parameters:

sockh is the socket to use for the call

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
long ids_addr_get_line_len (
    long sockh,          // Long in
    long * max_len       // Long out
);
```

Parameters:

sockh is the socket to use for the call

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
long ids_addr_get_option (
    long sockh,           // Long in
    char * param,         // String in
    char * value,         // String out
    long value_size
);
```

Parameters:

sockh is the socket to use for the call

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

ids_addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

```
long ids_addr_info (
    long sockh,           // Long in
    char * controls,      // String in
    char * value,         // String out
    long value_size
);
```

Parameters:

sockh is the socket to use for the call

controls this field contains the request information. It must be specified in the form ITEM=[value].

value this field contains the requested information.

Return Code:

negative for error, 0 for success

ids_addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

```
long ids_addr_init (
    long sockh,          // Long in
    long max_memory      // Long in
);
```

Parameters:

sockh is the socket to use for the call

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

ids_addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
long ids_addr_parse (
    long sockh,          // Long in
    long * field_length, // LongArray out
    long field_length_num
);
```

Parameters:

sockh is the socket to use for the call

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

ids_addr_preload_country

Description:

Use this function to preload country database

Prototype:

```
long ids_addr_preload_country (
    long    sockh,          // Long in
    char *   preload_type,   // String in
    char *   preload_country, // String in
    char *   val_mode        // String in
);
```

Parameters:

sockh is the socket to use for the call

preload_type Type of preload to perform

preload_country Country database to be preloaded

val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

ids_addr_set_attrib

Description:

Use this function to specify the character set of the data (for both input and output). The `default_country` parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

```
long ids_addr_set_attrib (
    long    sockh,          // Long in
    char *   char_set,       // String in
    char *   default_country // String in
);
```

Parameters:

sockh is the socket to use for the call

char_set The name of the character set used to encode the input and output.

default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

ids_addr_set_del_lines

Description:

Use this function to set delivery address line information

Prototype:

```
long ids_addr_set_del_lines (
    long    sockh,          // Long in
    char *  del_line1,      // Block in
    long    del_line1_size,
    char *  del_line2,      // Block in
    long    del_line2_size,
    char *  del_line3,      // Block in
    long    del_line3_size,
    char *  del_line4,      // Block in
    long    del_line4_size,
    char *  del_line5,      // Block in
    long    del_line5_size,
    char *  del_line6,      // Block in
    long    del_line6_size
);
```

Parameters:

sockh is the socket to use for the call

del_line1 delivery address line 1 input string

del_line2 delivery address line 2 input string

del_line3 delivery address line 3 input string

del_line4 delivery address line 4 input string

del_line5 delivery address line 5 input string

del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

ids_addr_set_field_case

Description:

Use this function to set individual input fields case option

Prototype:

```
long ids_addr_set_field_case (
    long    sockh,          // Long in
    long    field_idx,      // Long in
    long    field_case      // Long in
);
```

Parameters:

sockh is the socket to use for the call

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

ids_addr_set_field_ext

Description:

Use this function to set fields

Prototype:

```
long ids_addr_set_field_ext (
    long    sockh,          // Long in
    long    field_operation, // Long in
    char *  field_name,     // String in
    long    field_item_line, // Long in
    char *  field_type,     // String in
    char *  field_value,    // Block in
    long    field_value_size
);
```

Parameters:

sockh is the socket to use for the call

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

ids_addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
long ids_addr_set_field_idx (
    long    sockh,          // Long in
    long    field_idx,      // Long in
    char *  field_value,    // Block in
    long    field_value_size
);
```

Parameters:

sockh is the socket to use for the call

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

ids_addr_set_field_name

Description:

Use this function to set individual input fields by name

Prototype:

```
long ids_addr_set_field_name (
    long    sockh,          // Long in
    char *  field_name,     // String in
    char *  field_value,    // Block in
    long    field_value_size
);
```

Parameters:

sockh is the socket to use for the call

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

ids_addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
long ids_addr_set_lines (
    long    sockh,           // Long in
    char *  line_1,          // Block in
    long    line_1_size,
    char *  line_2,          // Block in
    long    line_2_size,
    char *  line_3,          // Block in
    long    line_3_size,
    char *  line_4,          // Block in
    long    line_4_size,
    char *  line_5,          // Block in
    long    line_5_size,
    char *  line_6,          // Block in
    long    line_6_size,
    char *  line_7,          // Block in
    long    line_7_size,
    char *  line_8,          // Block in
    long    line_8_size,
    char *  line_9,          // Block in
    long    line_9_size,
    char *  line_10,         // Block in
    long    line_10_size
);
```

Parameters:

sockh is the socket to use for the call

line_1 The first line of the address

line_2 The second line of the address

line_3 The third line of the address

line_4 The fourth line of the address

line_5 The fifth line of the address
line_6 The sixth line of the address
line_7 The seventh line of the address
line_8 The eighth line of the address
line_9 The ninth line of the address
line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

ids_addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
long ids_addr_set_option (
    long sockh,           // Long in
    char * param,         // String in
    char * value           // String in
);
```

Parameters:

sockh is the socket to use for the call

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

ids_addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable IDS Address Standardization Module to be installed.

Prototype:

```
long ids_addr_std (
    long sockh,           // Long in
    char * firm_name,     // String io
    long firm_name_size,
    char * urbanization,  // String io
    long urbanization_size,
    char * address_one,   // String io
    long address_one_size,
    char * address_two,   // String io
    long address_two_size,
    char * last_line,     // String io
    long last_line_size
);
```

Parameters:

sockh is the socket to use for the call

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

0 indicates an exact match to a valid address

1 indicates a no match (invalid address)

2 indicates a multi match (non-unique address), and

< 0 indicates an error

ids_addr_validate

Description:

Use this function to validate an address

Prototype:

```
long ids_addr_validate (
    long sockh,           // Long in
    long * status,        // Long out
    long * n_suggest      // Long out
);
```

Parameters:

sockh is the socket to use for the call

status The status returned by the validation process

n_suggest The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

ids_connect

Description:

Initiates a socket.

Prototype:

```
long ids_connect (
    char * host ,           // String in
    long port ,             // Long in
    long * sockh            // Long out
);
```

Parameters:

host is the host to connect to.

port is the port to connect to.

sockh is a socket handle.

Return Code:

negative for error, 0 for success

ids_disconnect

Description:

Releases resources allocated to a socket.

Prototype:

```
long ids_disconnect (
    long sockh              // Long in
);
```

Parameters:

sockh is the socket to use for the call

Return Code:

negative for error, 0 for success

ids_error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, **Error Log** section for help in interpreting the Error Log.

Prototype:

```
long ids_error_get (
    long    sockh,          // Long in
    char *  msg,            // String out
    long    msg_size
);
```

Parameters:

sockh is the socket to use for the call

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

ids_errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, **Error Log** section for information on interpreting the Error Log.

Prototype:

```
long ids_errors_get_all (
    long    sockh,          // Long in
    char *  msg,            // String out
    long    msg_size
);
```

Parameters:

sockh is the socket to use for the call

msg is an error message.

Return Code:

negative for error, 0 for success

ids_identify

Description:

Identify a session to the console

Prototype:

```
long ids_identify (
    long sockh,          // Long in
    char * identification // String in
);
```

Parameters:

sockh is the socket to use for the call

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

ids_is_little_endian

Description:

Checks if the search server is running on a little endian platform

Prototype:

```
long ids_is_little_endian (
    long sockh,          // Long in
    long * endian_state  // Long out
);
```

Parameters:

sockh is the socket to use for the call

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

ids_match_explain

Description:

Explains the match result given search and file records As `match_explain_count` does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

```
long ids_match_explain (
    long sockh,           // Long in
    char * search,        // String in
    char * match_tolerance, // String in
    char * searchrec,      // Block in
    long searchrec_size,
    char * filerec,        // Block in
    long filerec_size,
    char ** info_array,    // BlockArray out
    long info_array_num,
    long info_array_size
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the Search which was performed.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

ids_match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

```
long ids_match_explain_count (
    long sockh,          // Long in
    char * search,       // String in
    long * count         // Long out
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

ids_real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

```
long ids_real_time_async_get (
    long sockh,          // Long in
    char * reference,    // String in
    long block,          // Long in
    long * cluster_action_count // Long out
);
```

Parameters:

sockh is the socket to use for the call

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

ids_real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

```
long ids_real_time_async_start (
    long    sockh,           // Long in
    char *  rulebase,        // String in
    char *  system,          // String in
    char *  IDT,             // String in
    char *  sequence_number, // String in
    char *  operation,       // String in
    char *  cluster_record,  // Block in
    long    cluster_record_size,
    long    source,          // Long in
    char *  multi_search,    // String in
    long    input_id,        // Long in
    char *  reference,       // String out
    long    reference_size
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

ids_real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
long ids_real_time_flul_add (
    long    sockh,           // Long in
    char *  rule_type,       // String in
    char *  subject_rec_pk,  // Block in
    long    subject_rec_pk_size,
    char *  relationship,    // String in
    char *  related_rec_pk,  // Block in
    long    related_rec_pk_size
);
```

Parameters:

sockh is the socket to use for the call

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

ids_real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Prototype:

```
long ids_real_time_flul_close (
    long    sockh           // Long in
);
```

Parameters:

sockh is the socket to use for the call

Return Code:

negative for error, 0 for success

ids_real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
long ids_real_time_flul_delete (
    long    sockh,           // Long in
    long    rule_type_option, // Long in
    char *  record_pk,       // Block in
    long    record_pk_size
);
```

Parameters:

sockh is the socket to use for the call

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

ids_real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
long ids_real_time_flul_find_rule (
    long    sockh,          // Long in
    char *  idt_rec,        // Block in
    long    idt_rec_size,
    long    option          // Long in
);
```

Parameters:

sockh is the socket to use for the call

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
long ids_real_time_flul_get_rule (
    long    sockh,          // Long in
    char *  idt_rec,        // Block out
    long    idt_rec_size
);
```

Parameters:

sockh is the socket to use for the call

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

```
long ids_real_time_flul_init (
    long    sockh,           // Long in
    char *  idt_name,        // String in
    char *  multi_search     // String in
);
```

Parameters:

sockh is the socket to use for the call

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

ids_real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

```
long ids_real_time_sync_get (
    long    sockh,           // Long in
    char *  reference,       // String in
    char *  cluster_action_type, // String out
    long    cluster_action_type_size,
    char *  cluster_action_id, // String out
    long    cluster_action_id_size,
    long *  cluster_action_number, // Long out
    char *  cluster_action_new, // String out
    long    cluster_action_new_size
);
```

Parameters:

sockh is the socket to use for the call

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

ids_real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

```
long ids_real_time_sync_start (
    long    sockh,           // Long in
    char *  rulebase,        // String in
    char *  system,          // String in
    char *  IDT,             // String in
    char *  sequence_number, // String in
    char *  operation,       // String in
    char *  cluster_record,  // Block in
    long    cluster_record_size,
    long *  cluster_action_count, // Long out
    char *  reference,       // String out
    long    reference_size
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and positive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

ids_scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an Accept limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a SORT= parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

```
long ids_scores_get (
    long    sockh,           // Long in
    char *  searchname,     // String in
    long *  scores,         // LongArray out
    long    scores_num
);
```

Parameters:

sockh is the socket to use for the call

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

ids_search_comment_get

Description:

Returns the user defined comment stored with the search.

Prototype:

```
long ids_search_comment_get (
    long    sockh ,           // Long in
    char *  searchname ,     // String in
    char *  comment ,        // String out
    long    comment_size
);
```

Parameters:

sockh is the socket to use for the call

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

ids_search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

```

long ids_search_dedupe_start (
    long sockh,           // Long in
    char * search,        // String in
    char * search_width,  // String in
    char * match_tolerance, // String in
    char ** parameters,   // BlockArray in
    long parameters_num,
    long parameters_size,
    char * searchrec,      // Block io
    long searchrec_size,
    char * AnswersetName,  // String in
    long flags,            // Long in
    long * recid,          // Long io
    long * recs            // Long io
);

```

Parameters:

sockh is the socket to use for the call

search is the name of the search that is to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the `AnswersetName` parameter with the Search-Record-Id (10 bytes) . The maximum `AnswersetName` length is 22 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the `recid` of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the `recid` of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

ids_search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
long ids_search_fields_count (
    long sockh,          // Long in
    char * searchname,    // String in
    long * fc             // Long out
);
```

Parameters:

sockh is the socket to use for the call

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

ids_search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
long ids_search_fields_get (
    long sockh,          // Long in
    char * searchname,    // String in
    char ** fieldnames,    // StringArray out
    long fieldnames_num,
    long fieldnames_size
);
```

Parameters:

sockh is the socket to use for the call

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

ids_search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the **SQL Filters** section of the DESIGNER MANUAL for details about SQL filters

Prototype:

```
long ids_search_filter (
    long sockh,           // Long in
    char * search,        // String in
    char * filter         // String in
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

ids_search_finish

Description:

Release resources associated with `ids_search_start`.

Prototype:

```
long ids_search_finish (
    long sockh,           // Long in
    char * search         // String in
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

ids_search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

```
long ids_search_get (
    long sockh,          // Long in
    char * searchname,    // String in
    char * searchreturn,  // Block out
    long searchreturn_size,
    long * score,         // Long out
    long * sreps,         // LongArray out
    long sreps_num,
    long * freps,         // LongArray out
    long freps_num
);
```

Parameters:

sockh is the socket to use for the call

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and `SSA-NAME3 v1` is used.

Prototype:

```
long ids_search_get_complete (
    long    sockh,          // Long in
    char *  search,         // String in
    char *  searchreturn,   // Block out
    long    searchreturn_size,
    long *  score,          // Long out
    char *  info,           // Block out
    long    info_size
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4 \times (1 + 3 \times 100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

```
long ids_search_get_detail (
    long    sockh,          // Long in
    char *  search,         // String in
    char *  file_rec,       // Block out
    long    file_rec_size,
    long *  score,          // Long out
    char *  decision,       // String out
    long    decision_size,
    long *  file_recid      // Long out
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Prototype:

```
long ids_search_IDT_get (
    long    sockh,           // Long in
    char *  searchname,      // String in
    char *  IDT,             // String out
    long    IDT_size
);
```

Parameters:

sockh is the socket to use for the call

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

ids_search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

```
long ids_search_layout (
    long sockh,           // Long in
    char * search,        // String in
    char * viewType,      // String in
    char * func,          // String in
    char ** names,        // StringArray out
    long names_num,
    long names_size,
    long * lengths,       // LongArray out
    long lengths_num,
    long * offsets,       // LongArray out
    long offsets_num,
    long * repeats,       // LongArray out
    long repeats_num,
    char ** formats,      // StringArray out
    long formats_num,
    long formats_size
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the FORMATS_SIZE constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

ids_search_profile_count

Description:

Count the number of profile entries available

Prototype:

```
long ids_search_profile_count (
    long sockh,          // Long in
    long * count          // Long out
);
```

Parameters:

sockh is the socket to use for the call

count Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

```
long ids_search_profile_get (
    long sockh,          // Long in
    long * ids,          // LongArray out
    long ids_num,
    char ** names,       // StringArray out
    long names_num,
    long names_size,
    long * times,        // LongArray out
    long times_num
);
```

Parameters:

sockh is the socket to use for the call

ids the numeric identifiers for each field, valid values are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

```
long ids_search_start (
    long    sockh,           // Long in
    char *  search,          // String in
    char *  search_width,    // String in
    char *  match_tolerance, // String in
    char ** parameters,      // BlockArray in
    long    parameters_num,
    long    parameters_size,
    char *  searchrec,       // Block io
    long    searchrec_size,
    char *  AnswersetName,   // String in
    long *  recs,            // Long out
    char ** records,         // BlockArray in
    long    records_num,
    long    records_size
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an `AnswerSet`. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an `AnswerSet`, set `AnswersetName` to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding `SEARCH_LIMIT` and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

ids_search_start_via_parameters

Description:

Prototype:

```
long ids_search_start_via_parameters (
    long    sockh,           // Long in
    char *  search,          // String in
    char *  search_width,   // String in
    char *  match_tolerance, // String in
    char ** parameters,     // BlockArray in
    long    parameters_num,
    long    parameters_size,
    long *  datalen,         // Long out
    long *  recs             // Long out
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_start_via_record

Description:

Prototype:

```
long ids_search_start_via_record (
    long    sockh,           // Long in
    char *  search,          // String in
    char *  search_width,    // String in
    char *  match_tolerance, // String in
    char *  searchrec,       // Block in
    long    searchrec_size,
    long *  datalen,         // Long out
    long *  recs             // Long out
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

data**len** will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than **recs** records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
long ids_search_tolerances_count (
    long sockh,           // Long in
    char * searchname,    // String in
    long * count          // Long out
);
```

Parameters:

sockh is the socket to use for the call

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```

long ids_search_tolerances_get (
    long    sockh,          // Long in
    char *  searchname,     // String in
    char ** tolerances,     // StringArray out
    long    tolerances_num,
    long    tolerances_size
);

```

Parameters:

sockh is the socket to use for the call

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Prototype:

```

long ids_search_view_get (
    long    sockh,          // Long in
    char *  search,         // String in
    char *  viewType,       // String in
    char *  viewName,       // String out
    long    viewName_size,
    long *  viewFieldCount, // Long out
    long *  viewRecLen      // Long out
);

```

Parameters:

sockh is the socket to use for the call

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

ids_search_view_set

Description:

Sets a view as the active input or output view

Prototype:

```
long ids_search_view_set (
    long sockh,           // Long in
    char * search,        // String in
    char * viewType,      // String in
    char * viewName       // String in
);
```

Parameters:

sockh is the socket to use for the call

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

ids_search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
long ids_search_widths_count (
    long sockh,           // Long in
    char * searchname,    // String in
    long * count          // Long out
);
```

Parameters:

sockh is the socket to use for the call

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
long ids_search_widths_get (
    long    sockh,           // Long in
    char *   searchname,     // String in
    char **  widths,         // StringArray out
    long     widths_num,
    long     widths_size
);
```

Parameters:

sockh is the socket to use for the call

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_server_version_get

Description:

Get the version information associated with the server.

Prototype:

```
long ids_server_version_get (
    long    sockh,           // Long in
    char *   server_version, // String out
    long     server_version_size
);
```

Parameters:

sockh is the socket to use for the call

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

ids_session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
long ids_session_close (
    long sockh           // Long in
);
```

Parameters:

sockh is the socket to use for the call

Return Code:

negative for error, 0 for success

ids_session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
long ids_session_open (
    long sockh,           // Long in
    long * session        // Long io
);
```

Parameters:

sockh is the socket to use for the call

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

ids_set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the *Globalization* section of the OPERATIONS manual for further details and restrictions.

Prototype:

```
long ids_set_encoding (
    long sockh,          // Long in
    long encoding        // Long in
);
```

Parameters:

sockh is the socket to use for the call

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

ids_set_timeout

Description:

Informs the Search Server of the timeout value in seconds The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached)

Prototype:

```
long ids_set_timeout (
    long sockh,          // Long in
    long timeout         // Long in
);
```

Parameters:

sockh is the socket to use for the call

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

ids_set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, **VPD** section for details about VPD.

Prototype:

```
long ids_set_vpd_user (
    long    sockh,           // Long in
    char *  vpd_user,        // String in
    char *  vpd_ctx          // String in
);
```

Parameters:

sockh is the socket to use for the call

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

ids_system_close

Description:

closes the system and frees any remaining resources.

Prototype:

```
long ids_system_close (
    long    sockh           // Long in
);
```

Parameters:

sockh is the socket to use for the call

Return Code:

negative for error, 0 for success

ids_system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

```
long ids_system_idtname_count (
    long sockh,          // Long in
    long * idtcount      // Long out
);
```

Parameters:

sockh is the socket to use for the call

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

ids_system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

```
long ids_system_idtname_get (
    long sockh,          // Long in
    char ** idtnames,    // StringArray out
    long idtnames_num,
    long idtnames_size
);
```

Parameters:

sockh is the socket to use for the call

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_system_notify

Description:

Notifies search server on a system.

Prototype:

```
long ids_system_notify (
    long    sockh,           // Long in
    char *  rulebase,        // String in
    char *  sysname,         // String in
    char *  message          // String in
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

ids_system_open

Description:

opens a system.

Prototype:

```
long ids_system_open (
    long    sockh,           // Long in
    char *  rulebase,        // String in
    char *  system,          // String in
    char *  verbosity,       // String in
    char *  Options          // String in
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

ids_system_search_finish

Description:

Finishes the search and closes the system.

Prototype:

```
long ids_system_search_finish (
    long sockh           // Long in
);
```

Parameters:

sockh is the socket to use for the call

Return Code:

negative for error, 0 for success

ids_system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

```
long ids_system_search_start (
    long sockh,           // Long in
    char * rulebase,      // String in
    char * system,        // String in
    char * verbosity,     // String in
    char * options,       // String in
    char * search,        // String in
    char ** parameters,   // BlockArray in
    long parameters_num,
    long parameters_size,
    char * AnswersetName, // String in
    long * datalen,       // Long out
    long * recs           // Long out
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1, KEYWORD2=VALUE2,

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

datalen will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

ids_system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the SHOWALLSEARCHES keyword to the option string of `ids_system_open`.

Prototype:

```
long ids_system_searches_count (
    long sockh,          // Long in
    long * searchcount    // Long out
);
```

Parameters:

sockh is the socket to use for the call

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

ids_system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
long ids_system_searches_get (
    long sockh,          // Long in
    char ** searches,    // StringArray out
    long searches_num,
    long searches_size
);
```

Parameters:

sockh is the socket to use for the call

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_systems_count

Description:

the number of systems in the rulebase.

Prototype:

```
long ids_systems_count (
    long sockh,          // Long in
    char * rulebase,      // String in
    long * systemscount   // Long out
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

ids_systems_get

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

```
long ids_systems_get (
    long sockh,          // Long in
    char * rulebase,      // String in
    char ** systems,      // StringArray out
    long systems_num,
    long systems_size
);
```

Parameters:

sockh is the socket to use for the call

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Calling from C without Arrays

If the language does not support C-style arrays an alternative API is available. This API requires only the `long`, `long *` and `char *` data-types.

For Win32, the C API functions are provided by the import library `%SSALIB%\stssasea.lib` for dynamic linking.

For Unix, the C API functions are provided in `$SSABIN (libssasea.so and libssaio.k.so)`.

The functions are prototyped in `%SSAINC%\ssaseca.h`.

The functions return a response code. A negative response code indicates a transport error, after which the communication channel is closed and no further API calls can be made without reconnecting.

Constants

Constants are declared as `#defines`, in uppercase using underscores. Constants are prefixed with 'SSA_', eg. `SSA_MSG_SIZE`.

Parameter types

Note: values in **BOLD** represent information that must be provided to the function.

ids_addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
long ids_addr_get_cass_field (
    long sockh,
    long suggest_idx,    // Long in
    long field_idx,      // Long in
    char * field_value,  // Block out
    long field_value_size
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
long ids_addr_get_cass_field_cnt (
    long sockh,
    long * count           // Long out
);
```

Parameters:

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
long ids_addr_get_cass_field_info (
    long sockh,
    long suggest_idx,      // Long in
    long * field_length,   // LongArray out
    long field_length_num
);
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

ids_addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Prototype:

```
long ids_addr_get_del_lines_ext (
    long sockh,
    long suggest_idx,    // Long in
    long del_case,       // Long in
    char * del_line1,    // Block out
    long del_line1_size,
    char * del_line2,    // Block out
    long del_line2_size,
    char * del_line3,    // Block out
    long del_line3_size,
    char * del_line4,    // Block out
    long del_line4_size,
    char * del_line5,    // Block out
    long del_line5_size,
    char * del_line6,    // Block out
    long del_line6_size
);
```

Parameters:

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

ids_addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```
long ids_addr_get_field (
    long sockh,
    long suggest_idx,    // Long in
    long field_idx,      // Long in
    char * field_value,  // Block out
    long field_value_size,
    long * field_val_status, // Long out
    long * field_val_mods  // Long out
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

ids_addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
long ids_addr_get_field_count (
    long sockh,
    long * count          // Long out
);
```

Parameters:

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

ids_addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Prototype:

```
long ids_addr_get_field_ext (
    long sockh,
    long suggest_idx, // Long in
    long field_operation, // Long in
    char * field_name, // String in
    long field_item_line, // Long in
    char * field_type, // String in
    char * field_value, // Block out
    long field_value_size
);
```

Parameters:

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

ids_addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
long ids_addr_get_field_idx (
    long sockh,
    long suggest_idx,    // Long in
    long field_idx,      // Long in
    char * field_value,  // Block out
    long field_value_size
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

ids_addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
long ids_addr_get_field_info_ext (
    long sockh,
    long suggest_idx,    // Long in
    long * field_length, // LongArray out
    long field_length_num,
    char * addr_label_encoded, // Block out
    long addr_label_encoded_size,
    char * addr_label_charset, // String out
    long addr_label_charset_size,
    long * score           // Long out
);
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

ids_addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

```
long ids_addr_get_field_len (
    long sockh,
    long * max_len           // Long out
);
```

Parameters:

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
long ids_addr_get_line_len (
    long sockh,
    long * max_len           // Long out
);
```

Parameters:

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
long ids_addr_get_option (
    long sockh,
    char * param,          // String in
    char * value,         // String out
    long value_size
);
```

Parameters:

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

ids_addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

```
long ids_addr_info (
    long sockh,
    char * controls,       // String in
    char * value,         // String out
    long value_size
);
```

Parameters:

controls this field contains the request information. It must be specified in the form ITEM=[value].

value this field contains the requested information.

Return Code:

negative for error, 0 for success

ids_addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

```
long ids_addr_init (
    long sockh,
    long max_memory    // Long in
);
```

Parameters:

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

ids_addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
long ids_addr_parse (
    long sockh,
    long * field_length,    // LongArray out
    long field_length_num
);
```

Parameters:

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

ids_addr_preload_country

Description:

Use this function to preload country database

Prototype:

```
long ids_addr_preload_country (
    long sockh,
    char * preload_type, // String in
    char * preload_country, // String in
    char * val_mode // String in
);
```

Parameters:

preload_type Type of preload to perform

preload_country Country database to be preloaded

val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

ids_addr_set_attrib

Description:

Use this function to specify the character set of the data (for both input and output). The default_country parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

```
long ids_addr_set_attrib (
    long sockh,
    char * char_set, // String in
    char * default_country // String in
);
```

Parameters:

char_set The name of the character set used to encode the input and output.

default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

ids_addr_set_del_lines

Description:

Use this function to set delivery address line information

Prototype:

```
long ids_addr_set_del_lines (
    long sockh,
    char * del_line1,      // Block in
    long del_line1_size,
    char * del_line2,      // Block in
    long del_line2_size,
    char * del_line3,      // Block in
    long del_line3_size,
    char * del_line4,      // Block in
    long del_line4_size,
    char * del_line5,      // Block in
    long del_line5_size,
    char * del_line6,      // Block in
    long del_line6_size
);
```

Parameters:

del_line1 delivery address line 1 input string

del_line2 delivery address line 2 input string

del_line3 delivery address line 3 input string

del_line4 delivery address line 4 input string

del_line5 delivery address line 5 input string

del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

ids_addr_set_field_case

Description:

Use this function to set individual input fields case option

Prototype:

```
long ids_addr_set_field_case (
    long sockh,
    long field_idx,      // Long in
    long field_case      // Long in
);
```

Parameters:

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

ids_addr_set_field_ext

Description:

Use this function to set fields

Prototype:

```
long ids_addr_set_field_ext (
    long sockh,
    long field_operation, // Long in
    char * field_name,    // String in
    long field_item_line, // Long in
    char * field_type,    // String in
    char * field_value,   // Block in
    long field_value_size
);
```

Parameters:

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

ids_addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
long ids_addr_set_field_idx (
    long sockh,
    long field_idx,      // Long in
    char * field_value,  // Block in
    long field_value_size
);
```

Parameters:

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

ids_addr_set_field_name

Description:

Use this function to set individual input fields by name

Prototype:

```
long ids_addr_set_field_name (
    long sockh,
    char * field_name,    // String in
    char * field_value,  // Block in
    long field_value_size
);
```

Parameters:

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

ids_addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
long ids_addr_set_lines (
    long sockh,
    char * line_1,          // Block in
    long line_1_size,
    char * line_2,          // Block in
    long line_2_size,
    char * line_3,          // Block in
    long line_3_size,
    char * line_4,          // Block in
    long line_4_size,
    char * line_5,          // Block in
    long line_5_size,
    char * line_6,          // Block in
    long line_6_size,
    char * line_7,          // Block in
    long line_7_size,
    char * line_8,          // Block in
    long line_8_size,
    char * line_9,          // Block in
    long line_9_size,
    char * line_10,         // Block in
    long line_10_size
);
```

Parameters:

line_1 The first line of the address
line_2 The second line of the address
line_3 The third line of the address
line_4 The fourth line of the address
line_5 The fifth line of the address
line_6 The sixth line of the address
line_7 The seventh line of the address
line_8 The eighth line of the address
line_9 The ninth line of the address
line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

ids_addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
long ids_addr_set_option (
    long sockh,
    char * param,          // String in
    char * value           // String in
);
```

Parameters:

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

ids_addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable IDS Address Standardization Module to be installed.

Prototype:

```
long ids_addr_std (
    long sockh,
    char * firm_name,      // String io
    long  firm_name_size,
    char * urbanization,   // String io
    long  urbanization_size,
    char * address_one,    // String io
    long  address_one_size,
    char * address_two,    // String io
    long  address_two_size,
    char * last_line,      // String io
    long  last_line_size
);
```

Parameters:

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

- 0 indicates an exact match to a valid address
- 1 indicates a no match (invalid address)
- 2 indicates a multi match (non-unique address), and
- < 0 indicates an error

ids_addr_validate

Description:

Use this function to validate an address

Prototype:

```
long ids_addr_validate (
    long sockh,
    long * status,          // Long out
    long * n_suggest       // Long out
);
```

Parameters:

status The status returned by the validation process

n_suggest The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

ids_connect

Description:

Initiates a socket.

Prototype:

```
long ids_connect (
    char * host ,           // String in
    long port ,             // Long in
    long * sockh            // Long out
);
```

Parameters:

host is the host to connect to.

port is the port to connect to.

sockh is a socket handle.

Return Code:

negative for error, 0 for success

ids_disconnect

Description:

Releases resources allocated to a socket.

Prototype:

```
long ids_disconnect (
    long sockh ,
);
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Prototype:

```
long ids_error_get (
    long sockh,
    char * msg,           // String out
    long msg_size
);
```

Parameters:

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

ids_errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for information on interpreting the Error Log.

Prototype:

```
long ids_errors_get_all (
    long sockh,
    char * msg,           // String out
    long msg_size
);
```

Parameters:

msg is an error message.

Return Code:

negative for error, 0 for success

ids_identify

Description:

Identify a session to the console

Prototype:

```
long ids_identify (
    long sockh,
    char * identification // String in
);
```

Parameters:

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

ids_is_little_endian

Description:

Checks if the search server is running on a little endian platform

Prototype:

```
long ids_is_little_endian (
    long sockh,
    long * endian_state // Long out
);
```

Parameters:

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

ids_match_explain

Description:

Explains the match result given search and file records As match_explain_count does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

```
long ids_match_explain (
    long sockh,
    char * search,          // String in
    char * match_tolerance, // String in
    char * searchrec,       // Block in
    long searchrec_size,
    char * filerec,         // Block in
    long filerec_size,
    char * info_array,      // BlockArray out
    long info_array_num,
    long info_array_size
);
```

Parameters:

search is the name of the Search which was performed.

match_tolerance specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

ids_match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

```
long ids_match_explain_count (
    long sockh,
    char * search,          // String in
    long * count            // Long out
);
```

Parameters:

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

ids_real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

```
long ids_real_time_async_get (
    long sockh,
    char * reference,      // String in
    long block,            // Long in
    long * cluster_action_count // Long out
);
```

Parameters:

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

ids_real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

```
long ids_real_time_async_start (
    long sockh,
    char * rulebase,       // String in
    char * system,         // String in
    char * IDT,            // String in
    char * sequence_number, // String in
    char * operation,      // String in
    char * cluster_record, // Block in
    long cluster_record_size,

```

```

    long    source ,           // Long in
    char *  multi_search ,    // String in
    long    input_id ,        // Long in
    char *  reference ,       // String out
    long    reference_size
);

```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

ids_real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
long ids_real_time_flul_add (
    long sockh,
    char * rule_type,      // String in
    char * subject_rec_pk, // Block in
    long  subject_rec_pk_size,
    char * relationship,   // String in
    char * related_rec_pk, // Block in
    long  related_rec_pk_size
);
```

Parameters:

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

ids_real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Prototype:

```
long ids_real_time_flul_close (
    long sockh,
);
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
long ids_real_time_flul_delete (
    long sockh,
    long rule_type_option, // Long in
    char * record_pk,      // Block in
    long record_pk_size
);
```

Parameters:

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

ids_real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
long ids_real_time_flul_find_rule (
    long sockh,
    char * idt_rec,          // Block in
    long idt_rec_size,
    long option              // Long in
);
```

Parameters:

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
long ids_real_time_flul_get_rule (
    long sockh,
    char * idt_rec,          // Block out
    long idt_rec_size
);
```

Parameters:

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

```
long ids_real_time_flul_init (
    long sockh,
    char * idt_name,        // String in
    char * multi_search     // String in
);
```

Parameters:

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

ids_real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

```
long ids_real_time_sync_get (
    long sockh ,
    char * reference ,           // String in
    char * cluster_action_type , // String out
    long cluster_action_type_size ,
    char * cluster_action_id ,  // String out
    long cluster_action_id_size ,
    long * cluster_action_number , // Long out
    char * cluster_action_new ,  // String out
    long cluster_action_new_size
);
```

Parameters:

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

ids_real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

```
long ids_real_time_sync_start (
    long sockh,
    char * rulebase,          // String in
    char * system,           // String in
    char * IDT,              // String in
    char * sequence_number,  // String in
    char * operation,        // String in
    char * cluster_record,   // Block in
    long cluster_record_size,
    long * cluster_action_count, // Long out
    char * reference,        // String out
    long reference_size
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and postive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was lessa than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

ids_scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an `Accept` limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a `SORT=` parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

```
long ids_scores_get (
    long sockh ,
    char * searchname ,      // String in
    long * scores ,          // LongArray out
    long scores_num
);
```

Parameters:

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

ids_search_comment_get

Description:

Returns the user defined comment stored with the search.

Prototype:

```
long ids_search_comment_get (
    long sockh ,
    char * searchname ,      // String in
    char * comment ,         // String out
    long comment_size
);
```

Parameters:

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

ids_search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

```
long ids_search_dedupe_start (
    long sockh ,
    char * search ,           // String in
    char * search_width ,    // String in
    char * match_tolerance , // String in
    char * parameters ,      // BlockArray in
    long parameters_num ,
    long parameters_size ,
    char * searchrec ,       // Block io
    long searchrec_size ,
    char * AnswersetName ,   // String in
    long flags ,             // Long in
    long * recid ,           // Long io
    long * recs              // Long io
);
```

Parameters:

search is the name of the search that is to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the `AnswersetName` parameter with the Search-Record-Id (10 bytes) . The maximum `AnswersetName` length is 22 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the `recid` of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the `recid` of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

ids_search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
long ids_search_fields_count (
    long sockh,
    char * searchname,    // String in
    long * fc             // Long out
);
```

Parameters:

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

ids_search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
long ids_search_fields_get (
    long sockh,
    char * searchname,    // String in
    char * fieldnames,    // StringArray out
    long fieldnames_num,
    long fieldnames_size
);
```

Parameters:

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

ids_search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the *SQL Filters* section of the DESIGNER MANUAL for details about SQL filters

Prototype:

```
long ids_search_filter (
    long sockh ,
    char * search ,           // String in
    char * filter             // String in
);
```

Parameters:

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

ids_search_finish

Description:

Release resources associated with `ids_search_start`.

Prototype:

```
long ids_search_finish (
    long sockh ,
    char * search           // String in
);
```

Parameters:

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

ids_search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

```
long ids_search_get (
    long sockh,
    char * searchname,    // String in
    char * searchreturn,  // Block out
    long searchreturn_size,
    long * score,         // Long out
    long * sreps,         // LongArray out
    long sreps_num,
    long * freps,         // LongArray out
    long freps_num
);
```

Parameters:

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: **sreps** and **freps** are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; `sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0`

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and SSA-NAME3 v1 is used.

Prototype:

```
long ids_search_get_complete (
    long sockh,
    char * search,          // String in
    char * searchreturn,    // Block out
    long searchreturn_size,
    long * score,           // Long out
    char * info,            // Block out
    long info_size
);
```

Parameters:

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4*(1+3*100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

```

long ids_search_get_detail (
    long sockh,
    char * search,          // String in
    char * file_rec,        // Block out
    long file_rec_size,
    long * score,           // Long out
    char * decision,        // String out
    long decision_size,
    long * file_recid       // Long out
);

```

Parameters:

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Prototype:

```

long ids_search_IDT_get (
    long sockh,
    char * searchname,      // String in
    char * IDT,             // String out
    long IDT_size
);

```

Parameters:

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

ids_search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

```
long ids_search_layout (
    long sockh,
    char * search,          // String in
    char * viewType,       // String in
    char * func,           // String in
    char * names,          // StringArray out
    long names_num,
    long names_size,
    long * lengths,        // LongArray out
    long lengths_num,
    long * offsets,        // LongArray out
    long offsets_num,
    long * repeats,        // LongArray out
    long repeats_num,
    char * formats,        // StringArray out
    long formats_num,
    long formats_size
);
```

Parameters:

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)

Characters 13 - 50: Reserved

Note: It is recommended that the `FORMATS_SIZE` constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

ids_search_profile_count

Description:

Count the number of profile entries available

Prototype:

```
long ids_search_profile_count (
    long sockh,
    long * count           // Long out
);
```

Parameters:

count Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

```
long ids_search_profile_get (
    long sockh,
    long * ids,           // LongArray out
    long ids_num,
    char * names,         // StringArray out
    long names_num,
    long names_size,
    long * times,         // LongArray out
    long times_num
);
```

Parameters:

ids the numeric identifiers for each field, valid values are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

```
long ids_search_start (
    long sockh,
    char * search,           // String in
    char * search_width,    // String in
    char * match_tolerance, // String in
    char * parameters,      // BlockArray in
    long parameters_num,
    long parameters_size,
    char * searchrec,       // Block io
    long searchrec_size,
    char * AnswersetName,   // String in
    long * recs,            // Long out
    char * records,         // BlockArray in
    long records_num,
    long records_size
);
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an `AnswerSet`. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an `AnswerSet`, set `AnswersetName` to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding `SEARCH_LIMIT` and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

ids_search_start_via_parameters

Description:

Prototype:

```
long ids_search_start_via_parameters (
    long sockh,
    char * search,           // String in
    char * search_width,    // String in
    char * match_tolerance, // String in
    char * parameters,      // BlockArray in
    long parameters_num,
    long parameters_size,
    long * datalen,          // Long out
    long * recs              // Long out
);
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_start_via_record

Description:

Prototype:

```
long ids_search_start_via_record (
    long sockh ,
    char * search ,           // String in
    char * search_width ,    // String in
    char * match_tolerance , // String in
    char * searchrec ,       // Block in
    long searchrec_size ,
    long * datalen ,         // Long out
    long * recs              // Long out
);
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than **recs** records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
long ids_search_tolerances_count (
    long sockh,
    char * searchname,    // String in
    long * count          // Long out
);
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
long ids_search_tolerances_get (
    long sockh,
    char * searchname,    // String in
    char * tolerances,    // StringArray out
    long tolerances_num,
    long tolerances_size
);
```

Parameters:

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Prototype:

```
long ids_search_view_get (
    long sockh,
    char * search,           // String in
    char * viewType,        // String in
    char * viewName,        // String out
    long viewName_size,
    long * viewFieldCount,  // Long out
    long * viewRecLen       // Long out
);
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

ids_search_view_set

Description:

Sets a view as the active input or output view

Prototype:

```
long ids_search_view_set (  
    long sockh,  
    char * search,          // String in  
    char * viewType,       // String in  
    char * viewName        // String in  
);
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

ids_search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
long ids_search_widths_count (  
    long sockh,  
    char * searchname,      // String in  
    long * count            // Long out  
);
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
long ids_search_widths_get (
    long sockh,
    char * searchname,      // String in
    char * widths,          // StringArray out
    long widths_num,
    long widths_size
);
```

Parameters:

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_server_version_get

Description:

Get the version information associated with the server.

Prototype:

```
long ids_server_version_get (
    long sockh,
    char * server_version, // String out
    long server_version_size
);
```

Parameters:

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

ids_session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
long ids_session_close (
    long sockh ,
);
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
long ids_session_open (
    long sockh ,
    long * session // Long io
);
```

Parameters:

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

ids_set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the **Globalization** section of the OPERATIONS manual for further details and restrictions.

Prototype:

```
long ids_set_encoding (
    long sockh,
    long encoding      // Long in
);
```

Parameters:

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

ids_set_timeout

Description:

Informs the Search Server of the timeout value in seconds The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached)

Prototype:

```
long ids_set_timeout (
    long sockh,
    long timeout      // Long in
);
```

Parameters:

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

ids_set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, *VPD* section for details about VPD.

Prototype:

```
long ids_set_vpd_user (
    long sockh,
    char * vpd_user,      // String in
    char * vpd_ctx        // String in
);
```

Parameters:

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

ids_system_close

Description:

closes the system and frees any remaining resources.

Prototype:

```
long ids_system_close (
    long sockh,
);
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

```
long ids_system_idtname_count (
    long sockh,
    long * idtcount           // Long out
);
```

Parameters:

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

ids_system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

```
long ids_system_idtname_get (
    long sockh,
    char * idtnames,          // StringArray out
    long idtnames_num,
    long idtnames_size
);
```

Parameters:

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_system_notify

Description:

Notifies search server on a system.

Prototype:

```
long ids_system_notify (
    long sockh,
    char * rulebase,          // String in
    char * sysname,          // String in
    char * message            // String in
);
```

Parameters:

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

ids_system_open

Description:

opens a system.

Prototype:

```
long ids_system_open (
    long sockh,
    char * rulebase,      // String in
    char * system,        // String in
    char * verbosity,     // String in
    char * Options        // String in
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

ids_system_search_finish

Description:

Finishes the search and closes the system.

Prototype:

```
long ids_system_search_finish (
    long sockh ,
);
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

```
long ids_system_search_start (
    long sockh ,
    char * rulebase ,           // String in
    char * system ,            // String in
    char * verbosity ,         // String in
    char * options ,           // String in
    char * search ,            // String in
    char * parameters ,        // BlockArray in
    long parameters_num ,
    long parameters_size ,
    char * AnswersetName ,     // String in
    long * datalen ,           // Long out
    long * recs                // Long out
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1, KEYWORD2=VALUE2,

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

data**len** will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

ids_system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
long ids_system_searches_count (
    long sockh,
    long * searchcount    // Long out
);
```

Parameters:

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

ids_system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
long ids_system_searches_get (
    long sockh,
    char * searches,    // StringArray out
    long searches_num,
    long searches_size
);
```

Parameters:

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_systems_count

Description:

the number of systems in the rulebase.

Prototype:

```
long ids_systems_count (
    long sockh,
    char * rulebase,      // String in
    long * systemscount   // Long out
);
```

Parameters:

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

ids_systems_get

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

```
long ids_systems_get (
    long sockh,
    char * rulebase,      // String in
    char * systems,       // StringArray out
    long systems_num,
    long systems_size
);
```

Parameters:

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Calling from C#

In C# the API methods are made available in the shareable assembly **ssasecs.dll**. This contains the classes `ids`, which contains the API, as well as `SSAAPIException` and `SSASocketException`, which are based on `ApplicationException`. All of these are in the `ssa` namespace.

The `ids` methods throw exceptions of the classes `SSAAPIException` and `SSASocketException` that must be caught. If `SSASocketException` is caught, the communication channel is closed and no further calls to the API can be made without reconnecting.

Installation - Win32 client

The shareable assembly can be installed in the global assembly cache with Microsoft's **gacutil** utility:

```
gacutil /i %SSABIN%\ssasecs.dll
```

You can then create applications that use it with:

```
csc /reference:%SSABIN%\ssasecs.dll myprog.cs
```

Constants

Constants are public properties of the `ids` class, in uppercase using underscores, eg. `ids.MSG_SIZE`.

Response code

The response code returned from certain calls is also a public property called `rc` i.e. `ids.rc`.

Parameter types

Note: values in **BOLD** represent information that must be provided to the function.

Constructor

Description:

Used to create a `ssa.ids` object. This object is then used to make the required API calls to the Search or Connection Server.

Prototype:

```
using ssa;

public ids (
    String      hostname,    // String in
    int         port         // long in
) throws SSAException;
```

Parameters:

hostname is the name or IP address of the computer where the IIR Server is running

port the port number of the IIR Server for which a socket connection is to be established

Returns:

object allocated on success; `SSAException` on error

addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
using ssa;

public byte [] addr_get_cass_field (
    int      suggest_idx,    // Long in
    int      field_idx      // Long in
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
using ssa;  
  
public int addr_get_cass_field_cnt ();
```

Parameters:

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
using ssa;  
  
public int [] addr_get_cass_field_info (  
    int suggest_idx // Long in  
);
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Prototype:

```
using ssa;

public struct addr_get_del_lines_ext_struct addr_get_del_lines_ext (
    int      suggest_idx,    // Long in
    int      del_case        // Long in
);
```

Parameters:

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```
using ssa;

public struct addr_get_field_struct addr_get_field (
    int      suggest_idx,    // Long in
    int      field_idx       // Long in
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the field_length array for the `ids_addr_parse` API.

Prototype:

```
using ssa;  
  
public int addr_get_field_count ();
```

Parameters:

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Prototype:

```
using ssa;  
  
public byte [] addr_get_field_ext (  
    int    suggest_idx,      // Long in  
    int    field_operation, // Long in  
    string field_name,       // String in  
    int    field_item_line, // Long in  
    string field_type        // String in  
);
```

Parameters:

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
using ssa;

public byte [] addr_get_field_idx (
    int    suggest_idx,    // Long in
    int    field_idx       // Long in
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
using ssa;

public struct addr_get_field_info_ext_struct addr_get_field_info_ext (
    int suggest_idx // Long in
);
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

```
using ssa;

public int addr_get_field_len ();
```

Parameters:

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
using ssa;  
  
public int addr_get_line_len ();
```

Parameters:

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
using ssa;  
  
public string addr_get_option (  
    string param          // String in  
);
```

Parameters:

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

```
using ssa;

public string addr_info (
    string controls          // String in
);
```

Parameters:

controls this field contains the request information. It must be specified in the form ITEM=[value].

value this field contains the requested information.

Return Code:

negative for error, 0 for success

addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

```
using ssa;

public void addr_init (
    int max_memory          // Long in
);
```

Parameters:

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
using ssa;  
  
public int [] addr_parse ();
```

Parameters:

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

addr_preload_country

Description:

Use this function to preload country database

Prototype:

```
using ssa;  
  
public void addr_preload_country (  
    string preload_type, // String in  
    string preload_country, // String in  
    string val_mode // String in  
);
```

Parameters:

preload_type Type of preload to perform

preload_country Country database to be preloaded

val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

addr_set_attrb

Description:

Use this function to specify the character set of the data (for both input and output). The default_country parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

```
using ssa;

public void addr_set_attrib (
    string char_set,      // String in
    string default_country // String in
);
```

Parameters:

char_set The name of the character set used to encode the input and output.

default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

addr_set_del_lines

Description:

Use this function to set delivery address line information

Prototype:

```
using ssa;

public void addr_set_del_lines (
    byte [] del_line1, // Block in
    byte [] del_line2, // Block in
    byte [] del_line3, // Block in
    byte [] del_line4, // Block in
    byte [] del_line5, // Block in
    byte [] del_line6  // Block in
);
```

Parameters:

del_line1 delivery address line 1 input string

del_line2 delivery address line 2 input string

del_line3 delivery address line 3 input string

del_line4 delivery address line 4 input string

del_line5 delivery address line 5 input string

del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

addr_set_field_case

Description:

Use this function to set individual input fields case option

Prototype:

```
using ssa ;

public void addr_set_field_case (
    int      field_idx ,      // Long in
    int      field_case      // Long in
);
```

Parameters:

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

addr_set_field_ext

Description:

Use this function to set fields

Prototype:

```
using ssa ;

public void addr_set_field_ext (
    int      field_operation , // Long in
    string   field_name ,     // String in
    int      field_item_line , // Long in
    string   field_type ,     // String in
    byte []  field_value      // Block in
);
```

Parameters:

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
using ssa;

public void addr_set_field_idx (
    int      field_idx,      // Long in
    byte []  field_value     // Block in
);
```

Parameters:

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

addr_set_field_name

Description:

Use this function to set individual input fields by name

Prototype:

```
using ssa;

public void addr_set_field_name (
    string  field_name,      // String in
    byte [] field_value      // Block in
);
```

Parameters:

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
using ssa ;

public void addr_set_lines (
    byte [] line_1 ,           // Block in
    byte [] line_2 ,           // Block in
    byte [] line_3 ,           // Block in
    byte [] line_4 ,           // Block in
    byte [] line_5 ,           // Block in
    byte [] line_6 ,           // Block in
    byte [] line_7 ,           // Block in
    byte [] line_8 ,           // Block in
    byte [] line_9 ,           // Block in
    byte [] line_10            // Block in
);
```

Parameters:

line_1 The first line of the address

line_2 The second line of the address

line_3 The third line of the address

line_4 The fourth line of the address

line_5 The fifth line of the address

line_6 The sixth line of the address

line_7 The seventh line of the address

line_8 The eighth line of the address

line_9 The ninth line of the address

line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
using ssa;

public void addr_set_option (
    string param,           // String in
    string value            // String in
);
```

Parameters:

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable IDS Address Standardization Module to be installed.

Prototype:

```
using ssa;

public void addr_std (
    ref string firm_name,      // String io
    ref string urbanization,   // String io
    ref string address_one,    // String io
    ref string address_two,    // String io
    ref string last_line       // String io
);
```

Parameters:

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

0 indicates an exact match to a valid address

1 indicates a no match (invalid address)

2 indicates a multi match (non-unique address), and

< 0 indicates an error

addr_validate

Description:

Use this function to validate an address

Prototype:

```
using ssa;  
  
public struct addr_validate_struct addr_validate ();
```

Parameters:

status The status returned by the validation process

n_suggest The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

disconnect

Description:

Releases resources allocated to a socket.

Prototype:

```
using ssa;  
  
public void disconnect ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Prototype:

```
using ssa;  
  
public string error_get ();
```

Parameters:

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for information on interpreting the Error Log.

Prototype:

```
using ssa;  
  
public string errors_get_all ();
```

Parameters:

msg is an error message.

Return Code:

negative for error, 0 for success

identify

Description:

Identify a session to the console

Prototype:

```
using ssa;  
  
public void identify (  
    string identification // String in  
);
```

Parameters:

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

is_little_endian

Description:

Checks if the search server is running on a little endian platform

Prototype:

```
using ssa;  
  
public int is_little_endian ();
```

Parameters:

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

match_explain

Description:

Explains the match result given search and file records As `match_explain_count` does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

```
using ssa;

public byte [][] match_explain (
    string search,           // String in
    string match_tolerance, // String in
    byte [] searchrec,       // Block in
    byte [] filerec          // Block in
);
```

Parameters:

search is the name of the Search which was performed.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

```
using ssa;

public int match_explain_count (
    string search // String in
);
```

Parameters:

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

```
using ssa;

public int real_time_async_get (
    string reference, // String in
    int block // Long in
);
```

Parameters:

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

```
using ssa ;

public string real_time_async_start (
    string rulebase ,           // String in
    string system ,            // String in
    string IDT ,               // String in
    string sequence_number ,    // String in
    string operation ,         // String in
    byte [] cluster_record ,    // Block in
    int source ,               // Long in
    string multi_search ,      // String in
    int input_id               // Long in
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
using ssa;

public void real_time_flul_add (
    string rule_type,      // String in
    byte [] subject_rec_pk, // Block in
    string relationship,   // String in
    byte [] related_rec_pk // Block in
);
```

Parameters:

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Prototype:

```
using ssa;

public void real_time_flul_close ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
using ssa;

public void real_time_flul_delete (
    int      rule_type_option, // Long in
    byte []  record_pk        // Block in
);
```

Parameters:

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
using ssa;

public void real_time_flul_find_rule (
    byte []  idt_rec, // Block in
    int      option   // Long in
);
```

Parameters:

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.
1 Unlink Rule.

Return Code:

negative for error, 0 for success

real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
using ssa;  
  
public byte [] real_time_flul_get_rule ();
```

Parameters:

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

```
using ssa;  
  
public void real_time_flul_init (  
    string idt_name,          // String in  
    string multi_search      // String in  
);
```

Parameters:

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

```
using ssa ;

public struct real_time_sync_get_struct real_time_sync_get (
    string reference // String in
);
```

Parameters:

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

```

using ssa;

public struct real_time_sync_start_struct real_time_sync_start (
    string rulebase,          // String in
    string system,            // String in
    string IDT,               // String in
    string sequence_number,   // String in
    string operation,         // String in
    byte [] cluster_record    // Block in
);

```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and positive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with `WARN_DUPLICATE_PK` sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an `Accept` limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a `SORT=` parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

```
using ssa ;

public int [] scores_get (
    string searchname    // String in
);
```

Parameters:

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

search_comment_get

Description:

Returns the user defined comment stored with the search.

Prototype:

```
using ssa ;

public string search_comment_get (
    string searchname    // String in
);
```

Parameters:

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

```
using ssa ;

public void search_dedupe_start (
    string search,           // String in
    string search_width,     // String in
    string match_tolerance,  // String in
    byte [][] parameters,   // BlockArray in
    ref byte [] searchrec,   // Block io
    string AnswersetName,   // String in
    int flags,               // Long in
    ref int recid,           // Long io
    ref int recs             // Long io
);
```

Parameters:

search is the name of the search that is to be performed.

search_width specifies either *Narrow*, *Typical* or *Exhaustive* to nominate how many candidates should be selected.

match_tolerance specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the AnswersetName parameter with the Search-Record-Id (10 bytes) . The maximum AnswersetName length is 22 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the *recid* of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the *recid* of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
using ssa ;

public int search_fields_count (
    string searchname    // String in
);
```

Parameters:

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
using ssa ;

public string [] search_fields_get (
    string searchname    // String in
);
```

Parameters:

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the *SQL Filters* section of the DESIGNER MANUAL for details about SQL filters

Prototype:

```
using ssa ;

public void search_filter (
    string search,          // String in
    string filter           // String in
);
```

Parameters:

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

search_finish

Description:

Release resources associated with `ids_search_start`.

Prototype:

```
using ssa ;

public void search_finish (
    string search           // String in
);
```

Parameters:

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

```
using ssa;

public struct search_get_struct search_get (
    string searchname    // String in
);
```

Parameters:

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and `SSA-NAME3 v1` is used.

Prototype:

```
using ssa;

public struct search_get_complete_struct search_get_complete (
    string search          // String in
);
```

Parameters:

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The **info** field has a length of 4*(1+3*100) bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the **ids_search_start** or **ids_system_search_start** function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

```
using ssa;

public struct search_get_detail_struct search_get_detail (
    string search          // String in
);
```

Parameters:

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Prototype:

```
using ssa ;

public string search_IDT_get (
    string searchname    // String in
);
```

Parameters:

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

```
using ssa ;

public struct search_layout_struct search_layout (
    string search ,           // String in
    string viewType ,        // String in
    string func               // String in
);
```

Parameters:

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the `FORMATS_SIZE` constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

search_profile_count

Description:

Count the number of profile entries available

Prototype:

```
using ssa ;  
  
public int search_profile_count ();
```

Parameters:

count Returns the number of elements

Return Code:

negative for error, 0 for success

search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

```
using ssa ;

public struct search_profile_get_struct search_profile_get ();
```

Parameters:

ids the numeric identifiers for each field, valid value are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

```
using ssa ;

public int search_start (
    string search ,           // String in
    string search_width ,     // String in
```

```

    string match_tolerance, // String in
    byte [][][] parameters, // BlockArray in
    ref byte [] searchrec, // Block io
    string AnswersetName, // String in
    byte [][][] records // BlockArray in
);

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an `AnswerSet`. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an `AnswerSet`, set `AnswersetName` to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding `SEARCH_LIMIT` and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

search_start_via_parameters

Description:

Prototype:

```

using ssa;

public struct search_start_via_parameters_struct search_start_via_parameters (
    string search, // String in
    string search_width, // String in

```

```

        string match_tolerance, // String in
        byte [][] parameters    // BlockArray in
    );

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

search_start_via_record

Description:

Prototype:

```

using ssa;

public struct search_start_via_record_struct search_start_via_record (
    string search,           // String in
    string search_width,     // String in
    string match_tolerance,  // String in
    byte [] searchrec        // Block in
);

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If **searchrec** is specified it will be used to search (provided no **parameters** are supplied). Alternatively if **parameters** are specified then the search will be on a record constructed on those parameters and returned to the user.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than **recs** records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
using ssa ;

public int search_tolerances_count (
    string searchname // String in
);
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
using ssa;

public string [] search_tolerances_get (
    string searchname    // String in
);
```

Parameters:

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Prototype:

```
using ssa;

public struct search_view_get_struct search_view_get (
    string search,          // String in
    string viewType        // String in
);
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

search_view_set

Description:

Sets a view as the active input or output view

Prototype:

```
using ssa;

public void search_view_set (
    string search,           // String in
    string viewType,        // String in
    string viewName         // String in
);
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
using ssa;

public int search_widths_count (
    string searchname       // String in
);
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
using ssa;  
  
public string [] search_widths_get (  
    string searchname    // String in  
);
```

Parameters:

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

server_version_get

Description:

Get the version information associated with the server.

Prototype:

```
using ssa;  
  
public string server_version_get ();
```

Parameters:

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
using ssa;  
  
public void session_close ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
using ssa;  
  
public void session_open (  
    ref int session          // Long io  
);
```

Parameters:

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the *Globalization* section of the OPERATIONS manual for further details and restrictions.

Prototype:

```
using ssa;  
  
public void set_encoding (  
    int encoding // Long in  
);
```

Parameters:

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

set_timeout

Description:

Informs the Search Server of the timeout value in seconds The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached)

Prototype:

```
using ssa;  
  
public void set_timeout (  
    int timeout // Long in  
);
```

Parameters:

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, *VPD* section for details about VPD.

Prototype:

```
using ssa;  
  
public void set_vpd_user (  
    string vpd_user,      // String in  
    string vpd_ctx        // String in  
);
```

Parameters:

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

system_close

Description:

closes the system and frees any remaining resources.

Prototype:

```
using ssa;  
  
public void system_close ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

```
using ssa;  
  
public int system_idtname_count ();
```

Parameters:

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

```
using ssa;  
  
public string [] system_idtname_get ();
```

Parameters:

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

system_notify

Description:

Notifies search server on a system.

Prototype:

```
using ssa;  
  
public void system_notify (  
    string rulebase,      // String in  
    string sysname,       // String in  
    string message        // String in  
);
```

Parameters:

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

system_open

Description:

opens a system.

Prototype:

```
using ssa ;

public void system_open (
    string rulebase ,           // String in
    string system ,             // String in
    string verbosity ,          // String in
    string Options               // String in
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

system_search_finish

Description:

Finishes the search and closes the system.

Prototype:

```
using ssa;  
  
public void system_search_finish ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

```
using ssa;  
  
public struct system_search_start_struct system_search_start (  
    string rulebase,          // String in  
    string system,           // String in  
    string verbosity,        // String in  
    string options,          // String in  
    string search,           // String in  
    byte [][] parameters,    // BlockArray in  
    string AnswersetName      // String in  
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1,KEYWORD2=VALUE2,

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

datalen will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
using ssa ;  
  
public int system_searches_count ();
```

Parameters:

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
using ssa ;  
  
public string [] system_searches_get ();
```

Parameters:

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

systems_count

Description:

the number of systems in the rulebase.

Prototype:

```
using ssa;

public int systems_count (
    string rulebase        // String in
);
```

Parameters:

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

systems_get

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

```
using ssa;

public string [] systems_get (
    string rulebase        // String in
);
```

Parameters:

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Calling from Cobol (z/OS)

IIR can be called from MVS Cobol under TSO, batch and/or CICS .

Static calls from Cobol are supported by including the copy book %SSAINC%\ssasecbs.cpy in the Working-Storage section of a program.

Dynamic calls from Cobol are supported by including the copy books

- ♦ **ssasecbw.cpy** in the Working-Storage section, and

- ♦ **ssasecbp.cpy** in the Procedure Division.

Two sample programs are provided. **idscob1.cob** demonstrates calls from TSO/batch and **idscob2.cob** is suitable for calls from CICS. These are built using the sample JCL, **clcob1.jcl** and **clcob2.jcl** respectively.

To use IIR from TSO/batch user applications you need to have Language Environment support. This is required to run Cobol programs compiled with the more recent LE/370 Cobol compilers.

Language Environment and TCP/IP (CICS Sockets) must be installed and enabled in CICS in order to use the IIR CICS API.

Constants

Constants are declared as variables in Working-Storage, in uppercase using hyphens. Constants are prefixed with 'SSA-', eg. SSA-MSG-SIZE.

Installation

Two load dataset files are provided in TSO Transmit format. The following two dataset files are not part of the standard release package and must be requested from Global Customer Support if required.

```
ssa.xmit.iss\ssaversion{}.load  
ssa.xmit.iss\ssaversion{}.cics.load
```

These datasets should be transferred to MVS datasets using the TSO Receive facility and named **SSA.IIR10.1.0.LOAD** and **SSA.IIR10.1.0.CICS.LOAD** respectively.

Installation (TSO/batch)

API programs must be linked with a stub called **CIC001** in order to resolve the IIR entry points. When using static calls from Cobol, the stub will dynamically load two other modules (**SSASEA** and **SSAI0K**) which must be present in the STEPLIB. These modules are found in **SSA.IIR10.1.0.LOAD**.

Installation (CICS) -static calls

If using static calls, API programs must be linked with a stub called `CIC001` in order to resolve the IIR entry points. When using static calls from Cobol, the stub will dynamically load two other modules (`SSASEA` and `CSSAIOK`) which must be defined to CICS. **Note:** these modules are different than those provided for the TSO/batch interface.

To install, add the dataset **SSA.IIR10.1.0.CICS.LOAD** to your CICS STEPLIB (DFHRPL).

Define `SSASEA` and `CSSAIOK` to your CICS environment.

Installation (CICS) - dynamic calls

To install, add the dataset **SSA.IIR10.1.0.CICS.LOAD** to your CICS STEPLIB (DFHRPL).

Define `CIC001`, `SSAN3FN`, `SSASEA` and `CSSAIOK` to your CICS environment.

Defining programs to CICS

Use the CEDA commands: (use a group defined on your CICS system).

```
CEDA DEF PROGRAM(CSSAIOK) GROUP(SSAGROUP)
CEDA INS PROGRAM(CSSAIOK) GROUP(SSAGROUP)
```

Repeat for other three programs

Example Program Definition

```
VIEW PROGRAM(CSSAIOK) GROUP(SSAGROUP)
OBJECT CHARACTERISTICS CICS RELEASE = 0610
CEDA View Program( CSSAIOK )
Program : CSSAIOK
Group : SSAGROUP
DEscription :
Language : C                      Cobol | Assembler | Le370 | C | Pli
RELoad : No                       No | Yes
RESident : No                     No | Yes
USAge :                          Normal | Transient
USElpacopy : No                   No | Yes
Status : Enabled                  Enabled | Disabled
RSl : 00                          0-24 | Public
CEdf : Yes                       Yes | No
DAtalocation : Any               Below | Any
EXEckey : User                   User | Cics
CONcurrency : Quasirent          Quasirent | Threadsafe
REMOTE ATTRIBUTES
DYnamic : No                     No | Yes
+ REMOTESystem :
```

Parameter types

The API functions return a response code. A negative response code indicates a transport error, after which the communication channel is closed and no further API calls can be made without reconnecting.

IDS-ADDR-GET-CASS-FIELD

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
CALL IDS-ADDR-GET-CASS-FIELD USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY VALUE      SUGGEST-IDX,
    BY VALUE      FIELD-IDX,
    BY REFERENCE  FIELD-VALUE,
    BY VALUE      FIELD-VALUE-SIZE
END CALL
```

Parameters:

SUGGEST-IDX Specifies the nth suggestion from which to get a cass field

FIELD-IDX Specifies a cass field within the nth suggestion

FIELD-VALUE The cass field value

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-CASS-FIELD-CNT

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
CALL IDS-ADDR-GET-CASS-FIELD-CNT USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  COUNT
END CALL
```

Parameters:

COUNT Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-CASS-FIELD-INFO

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
CALL IDS-ADDR-GET-CASS-FIELD-INFO USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY VALUE      SUGGEST-IDX,
      BY REFERENCE  FIELD-LENGTH,
      BY VALUE      FIELD-LENGTH-NUM
END CALL
```

Parameters:

SUGGEST-IDX Specifies the suggestion from which to retrieve information

FIELD-LENGTH An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-DEL-LINES-EXT

Description:

Use this function to retrieve delivery address line information

Prototype:

```
CALL IDS-ADDR-GET-DEL-LINES-EXT USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY VALUE      SUGGEST-IDX,
      BY VALUE      DEL-CASE,
      BY REFERENCE  DEL-LINE1,
      BY VALUE      DEL-LINE1-SIZE,
      BY REFERENCE  DEL-LINE2,
      BY VALUE      DEL-LINE2-SIZE,
      BY REFERENCE  DEL-LINE3,
      BY VALUE      DEL-LINE3-SIZE,
      BY REFERENCE  DEL-LINE4,
```

```

        BY VALUE      DEL-LINE4-SIZE,
        BY REFERENCE  DEL-LINE5,
        BY VALUE      DEL-LINE5-SIZE,
        BY REFERENCE  DEL-LINE6,
        BY VALUE      DEL-LINE6-SIZE
    END CALL

```

Parameters:

SUGGEST-IDX Specifies the suggestion from which to get delivery address lines

DEL-CASE Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

DEL-LINE1 delivery address line 1 output string

DEL-LINE2 delivery address line 2 output string

DEL-LINE3 delivery address line 3 output string

DEL-LINE4 delivery address line 4 output string

DEL-LINE5 delivery address line 5 output string

DEL-LINE6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-FIELD

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```

CALL IDS-ADDR-GET-FIELD USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY VALUE      SUGGEST-IDX,
    BY VALUE      FIELD-IDX,
    BY REFERENCE  FIELD-VALUE,
    BY VALUE      FIELD-VALUE-SIZE,
    BY REFERENCE  FIELD-VAL-STATUS,
    BY REFERENCE  FIELD-VAL-MODS
END CALL

```

Parameters:

SUGGEST-IDX Specifies the nth suggestion from which to get a field

FIELD-IDX Specifies a field within the nth suggestion

FIELD-VALUE The field value

FIELD-VAL-STATUS Specifies how this field matched the validation data

FIELD-VAL-MODS Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-FIELD-COUNT

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the field_length array for the `ids_addr_parse` API.

Prototype:

```
CALL IDS-ADDR-GET-FIELD-COUNT USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY REFERENCE    COUNT
END CALL
```

Parameters:

COUNT Returns the max number of address fields

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-FIELD-EXT

Description:

Use this function to retrieve all getter fields

Prototype:

```
CALL IDS-ADDR-GET-FIELD-EXT USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY VALUE        SUGGEST-IDX,
      BY VALUE        FIELD-OPERATION,
      BY REFERENCE    FIELD-NAME,
      BY VALUE        FIELD-ITEM-LINE,
```

```

        BY REFERENCE    FIELD-TYPE,
        BY REFERENCE    FIELD-VALUE,
        BY VALUE        FIELD-VALUE-SIZE
    END CALL

```

Parameters:

SUGGEST-IDX Specifies the suggestion from which to get fields

FIELD-OPERATION Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

FIELD-NAME Refer AD Result.dtd for field names

FIELD-ITEM-LINE Represent field line number or field item number

FIELD-TYPE Refer AD Result.dtd for field attribute Type

FIELD-VALUE Cleansed field output

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-FIELD-IDX

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```

    CALL IDS-ADDR-GET-FIELD-IDX USING
        BY REFERENCE    RC,
        BY VALUE        SOCKH,
        BY VALUE        SUGGEST-IDX,
        BY VALUE        FIELD-IDX,
        BY REFERENCE    FIELD-VALUE,
        BY VALUE        FIELD-VALUE-SIZE
    END CALL

```

Parameters:

SUGGEST-IDX Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

FIELD-IDX Specifies a field within the nth suggestion

FIELD-VALUE The field value

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-FIELD-INFO-EXT

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
CALL IDS-ADDR-GET-FIELD-INFO-EXT USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY VALUE      SUGGEST-IDX,
    BY REFERENCE  FIELD-LENGTH,
    BY VALUE      FIELD-LENGTH-NUM,
    BY REFERENCE  ADDR-LABEL-ENCODED,
    BY VALUE      ADDR-LABEL-ENCODED-SIZE,
    BY REFERENCE  ADDR-LABEL-CHARSET,
    BY VALUE      ADDR-LABEL-CHARSET-SIZE,
    BY REFERENCE  SCORE
END CALL
```

Parameters:

SUGGEST-IDX Specifies the suggestion from which to retrieve information

FIELD-LENGTH An array containing the length of each address field

ADDR-LABEL-ENCODED The returned label

ADDR-LABEL-CHARSET The character set used in the address label

SCORE The returned label's score

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-FIELD-LEN

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

```
CALL IDS-ADDR-GET-FIELD-LEN USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY REFERENCE    MAX-LEN
END CALL
```

Parameters:

MAX-LEN Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-LINE-LEN

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
CALL IDS-ADDR-GET-LINE-LEN USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY REFERENCE    MAX-LEN
END CALL
```

Parameters:

MAX-LEN Returns the max line length in bytes

Return Code:

negative for error, 0 for success

IDS-ADDR-GET-OPTION

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
CALL IDS-ADDR-GET-OPTION USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  PARAM,
    BY REFERENCE  VALUE,
    BY VALUE      VALUE-SIZE
END CALL
```

Parameters:

PARAM This field specifies the name of the option to get.

VALUE Returns the value of the option.

Return Code:

negative for error, 0 for success

IDS-ADDR-INFO

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

```
CALL IDS-ADDR-INFO USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  CONTROLS,
    BY REFERENCE  VALUE,
    BY VALUE      VALUE-SIZE
END CALL
```

Parameters:

CONTROLS this field contains the request information. It must be specified in the form `ITEM=[value]`.

VALUE this field contains the requested information.

Return Code:

negative for error, 0 for success

IDS-ADDR-INIT

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

```
CALL IDS-ADDR-INIT USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY VALUE      MAX-MEMORY
END CALL
```

Parameters:

MAX-MEMORY This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

IDS-ADDR-PARSE

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
CALL IDS-ADDR-PARSE USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  FIELD-LENGTH,
    BY VALUE      FIELD-LENGTH-NUM
END CALL
```

Parameters:

FIELD-LENGTH An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

IDS-ADDR-PRELOAD-COUNTRY

Description:

Use this function to preload country database

Prototype:

```
CALL IDS-ADDR-PRELOAD-COUNTRY USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  PRELOAD-TYPE,
    BY REFERENCE  PRELOAD-COUNTRY,
    BY REFERENCE  VAL-MODE
END CALL
```

Parameters:

PRELOAD-TYPE Type of preload to perform

PRELOAD-COUNTRY Country database to be preloaded

VAL-MODE Validation mode to be used

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-ATTRIB

Description:

Use this function to specify the character set of the data (for both input and output). The default_country parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

```
CALL IDS-ADDR-SET-ATTRIB USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  CHAR-SET,
    BY REFERENCE  DEFAULT-COUNTRY
END CALL
```

Parameters:

CHAR-SET The name of the character set used to encode the input and output.

DEFAULT-COUNTRY The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-DEL-LINES

Description:

Use this function to set delivery address line information

Prototype:

```
CALL IDS-ADDR-SET-DEL-LINES USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  DEL-LINE1,
    BY REFERENCE  DEL-LINE2,
    BY REFERENCE  DEL-LINE3,
    BY REFERENCE  DEL-LINE4,
    BY REFERENCE  DEL-LINE5,
    BY REFERENCE  DEL-LINE6
END CALL
```

Parameters:

DEL-LINE1 delivery address line 1 input string

DEL-LINE2 delivery address line 2 input string

DEL-LINE3 delivery address line 3 input string

DEL-LINE4 delivery address line 4 input string

DEL-LINE5 delivery address line 5 input string

DEL-LINE6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-FIELD-CASE

Description:

Use this function to set individual input fields case option

Prototype:

```
CALL IDS-ADDR-SET-FIELD-CASE USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY VALUE      FIELD-IDX,
    BY VALUE      FIELD-CASE
END CALL
```

Parameters:

FIELD-IDX Specifies the nth field to set

FIELD-CASE Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-FIELD-EXT

Description:

Use this function to set fields

Prototype:

```
CALL IDS-ADDR-SET-FIELD-EXT USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY VALUE        FIELD-OPERATION,
      BY REFERENCE    FIELD-NAME,
      BY VALUE        FIELD-ITEM-LINE,
      BY REFERENCE    FIELD-TYPE,
      BY REFERENCE    FIELD-VALUE
END CALL
```

Parameters:

FIELD-OPERATION Field operation Option 0 for AddressElements Option 1 for AddressLines

FIELD-NAME Refer AD Result.dtd for field names

FIELD-ITEM-LINE Represent field line number or field item number

FIELD-TYPE Refer AD Result.dtd for field attribute Type

FIELD-VALUE input field value

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-FIELD-IDX

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
CALL IDS-ADDR-SET-FIELD-IDX USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY VALUE      FIELD-IDX,
    BY REFERENCE  FIELD-VALUE
END CALL
```

Parameters:

FIELD-IDX Specifies the nth field to set

FIELD-VALUE Specifies a value for the nth field

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-FIELD-NAME

Description:

Use this function to set individual input fields by name

Prototype:

```
CALL IDS-ADDR-SET-FIELD-NAME USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  FIELD-NAME,
    BY REFERENCE  FIELD-VALUE
END CALL
```

Parameters:

FIELD-NAME Specifies the name of the field to set

FIELD-VALUE Specifies a value for the field

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-LINES

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
CALL IDS-ADDR-SET-LINES USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  LINE-1,
      BY REFERENCE  LINE-2,
      BY REFERENCE  LINE-3,
      BY REFERENCE  LINE-4,
      BY REFERENCE  LINE-5,
      BY REFERENCE  LINE-6,
      BY REFERENCE  LINE-7,
      BY REFERENCE  LINE-8,
      BY REFERENCE  LINE-9,
      BY REFERENCE  LINE-10
END CALL
```

Parameters:

LINE-1 The first line of the address

LINE-2 The second line of the address

LINE-3 The third line of the address

LINE-4 The fourth line of the address

LINE-5 The fifth line of the address

LINE-6 The sixth line of the address

LINE-7 The seventh line of the address

LINE-8 The eighth line of the address

LINE-9 The ninth line of the address

LINE-10 The tenth line of the address

Return Code:

negative for error, 0 for success

IDS-ADDR-SET-OPTION

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
CALL IDS-ADDR-SET-OPTION USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  PARAM,
      BY REFERENCE  VALUE
END CALL
```

Parameters:

PARAM This field specifies the name of the option to set.

VALUE This field specifies a value for the option.

Return Code:

negative for error, 0 for success

IDS-ADDR-STD

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable IDS Address Standardization Module to be installed.

Prototype:

```
CALL IDS-ADDR-STD USING
    BY REFERENCE RC,
    BY VALUE SOCKH,
    BY REFERENCE FIRM-NAME,
    BY VALUE FIRM-NAME-SIZE,
    BY REFERENCE URBANIZATION,
    BY VALUE URBANIZATION-SIZE,
    BY REFERENCE ADDRESS-ONE,
    BY VALUE ADDRESS-ONE-SIZE,
    BY REFERENCE ADDRESS-TWO,
    BY VALUE ADDRESS-TWO-SIZE,
    BY REFERENCE LAST-LINE,
    BY VALUE LAST-LINE-SIZE
END CALL
```

Parameters:

FIRM-NAME It contains the name of the firm (may be blank).

URBANIZATION this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

ADDRESS-ONE this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

ADDRESS-TWO this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

LAST-LINE this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

- 0 indicates an exact match to a valid address
- 1 indicates a no match (invalid address)
- 2 indicates a multi match (non-unique address), and
- < 0 indicates an error

IDS-ADDR-VALIDATE

Description:

Use this function to validate an address

Prototype:

```
CALL IDS-ADDR-VALIDATE USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  STATUS,
      BY REFERENCE  N-SUGGEST
END CALL
```

Parameters:

STATUS The status returned by the validation process

N-SUGGEST The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

IDS-CONNECT

Description:

Initiates a socket.

Prototype:

```
CALL IDS-CONNECT USING
      BY REFERENCE  RC,
      BY REFERENCE  HOST,
      BY VALUE      PORT,
      BY REFERENCE  SOCKH
END CALL
```

Parameters:

HOST is the host to connect to.

PORT is the port to connect to.

SOCKH is a socket handle.

Return Code:

negative for error, 0 for success

IDS-DISCONNECT

Description:

Releases resources allocated to a socket.

Prototype:

```
CALL IDS-DISCONNECT USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH
END CALL
```

Parameters:

none

Return Code:

negative for error, 0 for success

IDS-ERROR-GET

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Prototype:

```
CALL IDS-ERROR-GET USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  MSG,
      BY VALUE      MSG-SIZE
END CALL
```

Parameters:

MSG is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

IDS-ERRORS-GET-ALL

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for information on interpreting the Error Log.

Prototype:

```
CALL IDS-ERRORS-GET-ALL USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  MSG,
      BY VALUE      MSG-SIZE
END CALL
```

Parameters:

MSG is an error message.

Return Code:

negative for error, 0 for success

IDS-IDENTIFY

Description:

Identify a session to the console

Prototype:

```
CALL IDS-IDENTIFY USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  IDENTIFICATION
END CALL
```

Parameters:

IDENTIFICATION is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

IDS-IS-LITTLE-ENDIAN

Description:

Checks if the search server is running on a little endian platform

Prototype:

```
CALL IDS-IS-LITTLE-ENDIAN USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    ENDIAN-STATE
END CALL
```

Parameters:

ENDIAN-STATE Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

IDS-MATCH-EXPLAIN

Description:

Explains the match result given search and file records As match_explain_count does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

```
CALL IDS-MATCH-EXPLAIN USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    SEARCH,
    BY REFERENCE    MATCH-TOLERANCE,
    BY REFERENCE    SEARCHREC,
    BY REFERENCE    FILEREC,
    BY REFERENCE    INFO-ARRAY,
    BY VALUE        INFO-ARRAY-NUM,
    BY VALUE        INFO-ARRAY-SIZE
END CALL
```

Parameters:

SEARCH is the name of the Search which was performed.

MATCH-TOLERANCE specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates.

SEARCHREC is the original record that we searched on

FILEREC is the record that was returned by the search

INFO-ARRAY An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

IDS-MATCH-EXPLAIN-COUNT

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

```
CALL IDS-MATCH-EXPLAIN-COUNT USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY REFERENCE    SEARCH,
      BY REFERENCE    COUNT
END CALL
```

Parameters:

SEARCH is the name of the Search which was performed.

COUNT Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

IDS-REAL-TIME-ASYNC-GET

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

```
CALL IDS-REAL-TIME-ASYNC-GET USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    REFERENCE,
    BY VALUE        BLOCK,
    BY REFERENCE    CLUSTER-ACTION-COUNT
END CALL
```

Parameters:

REFERENCE A reference number identifying the request (returned by `ids_real_time_sync_start`)

BLOCK 1 = wait for a response 0 = return immediately if no results are available yet

CLUSTER-ACTION-COUNT The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

IDS-REAL-TIME-ASYNC-START

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

```
CALL IDS-REAL-TIME-ASYNC-START USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    RULEBASE,
    BY REFERENCE    SYSTEM,
    BY REFERENCE    IDT,
    BY REFERENCE    SEQUENCE-NUMBER,
    BY REFERENCE    OPERATION,
    BY REFERENCE    CLUSTER-RECORD,
    BY VALUE        SOURCE,
    BY REFERENCE    MULTI-SEARCH,
    BY VALUE        INPUT-ID,
    BY REFERENCE    REFERENCE,
    BY VALUE        REFERENCE-SIZE
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSTEM The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

SEQUENCE-NUMBER is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

OPERATION The synchronizer operation being performed: A for add, D for delete or U for update

CLUSTER-RECORD The record to be updated. The record must use the same layout as the IDT.

SOURCE Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

MULTI-SEARCH The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

INPUT-ID Reserved. A value of 0 must be passed for this parameter.

REFERENCE A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with `WARN_DUPLICATE_PK` sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

IDS-REAL-TIME-FLUL-ADD

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
CALL IDS-REAL-TIME-FLUL-ADD USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY REFERENCE    RULE-TYPE,
      BY REFERENCE    SUBJECT-REC-PK,
      BY REFERENCE    RELATIONSHIP,
      BY REFERENCE    RELATED-REC-PK
END CALL
```

Parameters:

RULE-TYPE This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

SUBJECT-REC-PK This field is for specifying the PK of the subject record

RELATIONSHIP This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

RELATED-REC-PK This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

IDS-REAL-TIME-FLUL-CLOSE

Description:

This API used to close and release force link and unlink module.

Prototype:

```
CALL IDS-REAL-TIME-FLUL-CLOSE USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH
END CALL
```

Parameters:

none

Return Code:

negative for error, 0 for success

IDS-REAL-TIME-FLUL-DELETE

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
CALL IDS-REAL-TIME-FLUL-DELETE USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY VALUE      RULE-TYPE-OPTION,
      BY REFERENCE  RECORD-PK
END CALL
```

Parameters:

RULE-TYPE-OPTION 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

RECORD-PK This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

IDS-REAL-TIME-FLUL-FIND-RULE

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
CALL IDS-REAL-TIME-FLUL-FIND-RULE USING
      BY REFERENCE      RC,
      BY VALUE           SOCKH,
      BY REFERENCE      IDT-REC,
      BY VALUE           OPTION
END CALL
```

Parameters:

IDT-REC This field is for specifying the PK of the record to be searched

OPTION 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

IDS-REAL-TIME-FLUL-GET-RULE

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
CALL IDS-REAL-TIME-FLUL-GET-RULE USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  IDT-REC,
    BY VALUE      IDT-REC-SIZE
END CALL
```

Parameters:

IDT-REC is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

IDS-REAL-TIME-FLUL-INIT

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

```
CALL IDS-REAL-TIME-FLUL-INIT USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  IDT-NAME,
    BY REFERENCE  MULTI-SEARCH
END CALL
```

Parameters:

IDT-NAME is the name of the IDT associated with the force link and unlink rule.

MULTI-SEARCH The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

IDS-REAL-TIME-SYNC-GET

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

```
CALL IDS-REAL-TIME-SYNC-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  REFERENCE,
    BY REFERENCE  CLUSTER-ACTION-TYPE,
    BY VALUE      CLUSTER-ACTION-TYPE-SIZE,
    BY REFERENCE  CLUSTER-ACTION-ID,
    BY VALUE      CLUSTER-ACTION-ID-SIZE,
    BY REFERENCE  CLUSTER-ACTION-NUMBER,
    BY REFERENCE  CLUSTER-ACTION-NEW,
    BY VALUE      CLUSTER-ACTION-NEW-SIZE
END CALL
```

Parameters:

REFERENCE A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

CLUSTER-ACTION-TYPE Identifies the action, ie add or delete

CLUSTER-ACTION-ID The prefix which identifies the cluster

CLUSTER-ACTION-NUMBER The number which identifies the cluster

CLUSTER-ACTION-NEW Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

IDS-REAL-TIME-SYNC-START

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

```
CALL IDS-REAL-TIME-SYNC-START USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  RULEBASE,
    BY REFERENCE  SYSTEM,
    BY REFERENCE  IDT,
    BY REFERENCE  SEQUENCE-NUMBER,
    BY REFERENCE  OPERATION,
    BY REFERENCE  CLUSTER-RECORD,
    BY REFERENCE  CLUSTER-ACTION-COUNT,
    BY REFERENCE  REFERENCE,
    BY VALUE      REFERENCE-SIZE
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSTEM The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

SEQUENCE-NUMBER is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

OPERATION The synchronizer operation being performed: A for add, D for delete or U for update

CLUSTER-RECORD The record to be updated. The record must use the same layout as the IDT.

CLUSTER-ACTION-COUNT The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

REFERENCE A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and positive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with `WARN_DUPLICATE_PK` sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

IDS-SCORES-GET

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an Accept limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a `SORT=` parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

```
CALL IDS-SCORES-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SEARCHNAME,
    BY REFERENCE  SCORES,
    BY VALUE      SCORES-NUM
END CALL
```

Parameters:

SEARCHNAME is the name of the associated search

SCORES is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

IDS-SEARCH-COMMENT-GET

Description:

Returns the user defined comment stored with the search.

Prototype:

```
CALL IDS-SEARCH-COMMENT-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SEARCHNAME,
    BY REFERENCE  COMMENT,
    BY VALUE      COMMENT-SIZE
END CALL
```

Parameters:

SEARCHNAME the search to count.

COMMENT is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

IDS-SEARCH-DEDUPE-START

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

```
CALL IDS-SEARCH-DEDUPE-START USING
    BY REFERENCE RC,
    BY VALUE SOCKH,
    BY REFERENCE SEARCH,
    BY REFERENCE SEARCH-WIDTH,
    BY REFERENCE MATCH-TOLERANCE,
    BY REFERENCE PARAMETERS,
    BY REFERENCE SEARCHREC,
    BY VALUE SEARCHREC-SIZE,
    BY REFERENCE ANSWERSETNAME,
    BY VALUE FLAGS,
    BY REFERENCE RECID,
    BY REFERENCE RECS
END CALL
```

Parameters:

SEARCH is the name of the search that is to be performed.

SEARCH-WIDTH specifies either *Narrow*, *Typical* or *Exhaustive* to nominate how many candidates should be selected.

MATCH-TOLERANCE specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates.

PARAMETERS not used.

SEARCHREC is the IDT record used to search with. It is returned by the Search Server.

ANSWERSETNAME is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the *AnswersetName* parameter with the *Search-Record-Id* (10 bytes) . The maximum *AnswersetName* length is 22 characters. If you do not wish to store the search results in an AnswerSet, set *AnswersetName* to an empty String.

FLAGS flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

RECID the *recid* of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the *recid* of the next record to be searched.

RECS the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

IDS-SEARCH-FIELDS-COUNT

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
CALL IDS-SEARCH-FIELDS-COUNT USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  SEARCHNAME,
      BY REFERENCE  FC
END CALL
```

Parameters:

SEARCHNAME the search to count.

FC is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

IDS-SEARCH-FIELDS-GET

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
CALL IDS-SEARCH-FIELDS-GET USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  SEARCHNAME,
      BY REFERENCE  FIELDNAMES,
      BY VALUE      FIELDNAMES-NUM,
      BY VALUE      FIELDNAMES-SIZE
END CALL
```

Parameters:

SEARCHNAME the search to count.

FIELDNAMES is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

IDS-SEARCH-FILTER

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the ***SQL Filters*** section of the DESIGNER MANUAL for details about SQL filters

Prototype:

```
CALL IDS-SEARCH-FILTER USING
    BY REFERENCE RC,
    BY VALUE SOCKH,
    BY REFERENCE SEARCH,
    BY REFERENCE FILTER
END CALL
```

Parameters:

SEARCH is the name of the Search that will use the filter

FILTER is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

IDS-SEARCH-FINISH

Description:

Release resources associated with `ids_search_start`.

Prototype:

```
CALL IDS-SEARCH-FINISH USING
    BY REFERENCE RC,
    BY VALUE SOCKH,
    BY REFERENCE SEARCH
END CALL
```

Parameters:

SEARCH is the name of the search that was performed

Return Code:

negative for error, 0 for success

IDS-SEARCH-GET

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

```
CALL IDS-SEARCH-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SEARCHNAME,
    BY REFERENCE  SEARCHRETURN,
    BY VALUE      SEARCHRETURN-SIZE,
    BY REFERENCE  SCORE,
    BY REFERENCE  SREPS,
    BY VALUE      SREPS-NUM,
    BY REFERENCE  FREPS,
    BY VALUE      FREPS-NUM
END CALL
```

Parameters:

SEARCHNAME is the name the search to used by the call.

SEARCHRETURN is an area into which a string from the set is copied.

SCORE is the score calculated for the string.

SREPS is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

FREPS is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

IDS-SEARCH-GET-COMPLETE

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and `SSA-NAME3 v1` is used.

Prototype:

```
CALL IDS-SEARCH-GET-COMPLETE USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    SEARCH,
    BY REFERENCE    SEARCHRETURN,
    BY VALUE        SEARCHRETURN-SIZE,
    BY REFERENCE    SCORE,
    BY REFERENCE    INFO,
    BY VALUE        INFO-SIZE
END CALL
```

Parameters:

SEARCH is the name of the search which was performed.

SEARCHRETURN is an area into which a string from the set is copied.

SCORE is the score calculated for the string.

INFO an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4 \times (1 + 3 \times 100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

IDS-SEARCH-GET-DETAIL

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

```
CALL IDS-SEARCH-GET-DETAIL USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    SEARCH,
    BY REFERENCE    FILE-REC,
    BY VALUE        FILE-REC-SIZE,
    BY REFERENCE    SCORE,
    BY REFERENCE    DECISION,
    BY VALUE        DECISION-SIZE,
    BY REFERENCE    FILE-RECID
END CALL
```

Parameters:

SEARCH is the name of the Search which was performed.

FILE-REC is the matched File record.

SCORE is the degree of similarity between the Search and File records (0-100).

DECISION is the match decision: A(ccept) or U(ndecided)

FILE-RECID is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

IDS-SEARCH-IDT-GET

Description:

Gets the name of the IDT associated with the search.

Prototype:

```
CALL IDS-SEARCH-IDT-GET USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  SEARCHNAME,
      BY REFERENCE  IDT,
      BY VALUE      IDT-SIZE
END CALL
```

Parameters:

SEARCHNAME is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

IDS-SEARCH-LAYOUT

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

```
CALL IDS-SEARCH-LAYOUT USING
    BY REFERENCE RC,
    BY VALUE SOCKH,
    BY REFERENCE SEARCH,
    BY REFERENCE VIEWTYPE,
    BY REFERENCE FUNC,
    BY REFERENCE NAMES,
    BY VALUE NAMES-NUM,
    BY VALUE NAMES-SIZE,
    BY REFERENCE LENGTHS,
    BY VALUE LENGTHS-NUM,
    BY REFERENCE OFFSETS,
    BY VALUE OFFSETS-NUM,
    BY REFERENCE REPEATS,
    BY VALUE REPEATS-NUM,
    BY REFERENCE FORMATS,
    BY VALUE FORMATS-NUM,
    BY VALUE FORMATS-SIZE
END CALL
```

Parameters:

SEARCH is the name of the Search which was performed.

VIEWTYPE the type of view: input or output.

FUNC describes the order of fields.

NAMES is the area into which an array containing the fieldnames will be copied.

LENGTHS is the area into which an array containing the lengths of the fields will be copied.

OFFSETS is the area into which an array containing the offsets of the fields will be copied.

REPEATS is the area into which an array containing the number of repeats in a field will be copied.

FORMATS is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters 2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the `FORMATS_SIZE` constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

IDS-SEARCH-PROFILE-COUNT

Description:

Count the number of profile entries available

Prototype:

```
CALL IDS-SEARCH-PROFILE-COUNT USING
      BY REFERENCE    RC,
      BY VALUE         SOCKH,
      BY REFERENCE     COUNT
END CALL
```

Parameters:

COUNT Returns the number of elements

Return Code:

negative for error, 0 for success

IDS-SEARCH-PROFILE-GET

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

```
CALL IDS-SEARCH-PROFILE-GET USING
      BY REFERENCE    RC,
      BY VALUE         SOCKH,
      BY REFERENCE     IDS ,
      BY VALUE         IDS-NUM,
      BY REFERENCE     NAMES,
      BY VALUE         NAMES-NUM,
      BY VALUE         NAMES-SIZE,
      BY REFERENCE     TIMES,
      BY VALUE         TIMES-NUM
END CALL
```

Parameters:

IDS the numeric identifiers for each field, valid value are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

NAMES the names for each field

TIMES the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

IDS-SEARCH-START

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

```
CALL IDS-SEARCH-START USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SEARCH,
    BY REFERENCE  SEARCH-WIDTH,
    BY REFERENCE  MATCH-TOLERANCE,
    BY REFERENCE  PARAMETERS,
    BY REFERENCE  SEARCHREC,
    BY VALUE      SEARCHREC-SIZE,
    BY REFERENCE  ANSWERSETNAME,
    BY REFERENCE  RECS,
    BY REFERENCE  RECORDS
END CALL
```

Parameters:

SEARCH is the name of the search to be performed.

SEARCH-WIDTH specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

MATCH-TOLERANCE specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

PARAMETERS contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

SEARCHREC is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

ANSWERSETNAME is used store the search results in an AnswerSet. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

RECS number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

RECORDS contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding `SEARCH_LIMIT` and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

IDS-SEARCH-START-VIA-PARAMETERS

Description:

Prototype:

```
CALL IDS-SEARCH-START-VIA-PARAMETERS USING
      BY REFERENCE      RC,
      BY VALUE          SOCKH,
      BY REFERENCE      SEARCH,
      BY REFERENCE      SEARCH-WIDTH,
      BY REFERENCE      MATCH-TOLERANCE,
      BY REFERENCE      PARAMETERS,
      BY REFERENCE      DATALEN,
      BY REFERENCE      RECS
END CALL
```

Parameters:

SEARCH is the name of the search to be performed.

SEARCH-WIDTH specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

MATCH-TOLERANCE specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

PARAMETERS contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

DATALEN will return the length of a record.

RECS number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

IDS-SEARCH-START-VIA-RECORD

Description:

Prototype:

```
CALL IDS-SEARCH-START-VIA-RECORD USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    SEARCH,
    BY REFERENCE    SEARCH-WIDTH,
    BY REFERENCE    MATCH-TOLERANCE,
    BY REFERENCE    SEARCHREC,
    BY REFERENCE    DATALEN,
    BY REFERENCE    RECS
END CALL
```

Parameters:

SEARCH is the name of the search to be performed.

SEARCH-WIDTH specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

MATCH-TOLERANCE specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

SEARCHREC is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no parameters are supplied). Alternatively if parameters are specified then the search will be on a record constructed on those parameters and returned to the user.

DATALEN will return the length of a record.

RECS number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

IDS-SEARCH-TOLERANCES-COUNT

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
CALL IDS-SEARCH-TOLERANCES-COUNT USING
      BY REFERENCE    RC,
      BY VALUE         SOCKH,
      BY REFERENCE    SEARCHNAME,
      BY REFERENCE    COUNT
END CALL
```

Parameters:

SEARCHNAME is the name the search to used by the call.

COUNT is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

IDS-SEARCH-TOLERANCES-GET

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
CALL IDS-SEARCH-TOLERANCES-GET USING
      BY REFERENCE    RC,
      BY VALUE         SOCKH,
      BY REFERENCE    SEARCHNAME,
      BY REFERENCE    TOLERANCES,
      BY VALUE         TOLERANCES-NUM,
      BY VALUE         TOLERANCES-SIZE
END CALL
```

Parameters:

SEARCHNAME the search to count.

TOLERANCES is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

IDS-SEARCH-VIEW-GET

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Prototype:

```
CALL IDS-SEARCH-VIEW-GET USING
    BY REFERENCE    RC,
    BY VALUE         SOCKH,
    BY REFERENCE    SEARCH,
    BY REFERENCE    VIEWTYPE,
    BY REFERENCE    VIEWNAME,
    BY VALUE         VIEWNAME-SIZE,
    BY REFERENCE    VIEWFIELDCOUNT,
    BY REFERENCE    VIEWRECLEN
END CALL
```

Parameters:

SEARCH is the name of the Search

VIEWTYPE the type of the view (input or output)

VIEWNAME is the name of the view to query

VIEWFIELDCOUNT the number of fields in the view

VIEWRECLEN is the length of the view

Return Code:

negative for error, 0 for success

IDS-SEARCH-VIEW-SET

Description:

Sets a view as the active input or output view

Prototype:

```
CALL IDS-SEARCH-VIEW-SET USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  SEARCH,
      BY REFERENCE  VIEWTYPE,
      BY REFERENCE  VIEWNAME
END CALL
```

Parameters:

SEARCH is the name of the Search

VIEWTYPE the type of the view (input or output)

VIEWNAME the name of the view to use

Return Code:

negative for error, 0 for success

IDS-SEARCH-WIDTHS-COUNT

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
CALL IDS-SEARCH-WIDTHS-COUNT USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  SEARCHNAME,
      BY REFERENCE  COUNT
END CALL
```

Parameters:

SEARCHNAME is the name the search to used by the call.

COUNT is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

IDS-SEARCH-WIDTHS-GET

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
CALL IDS-SEARCH-WIDTHS-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SEARCHNAME,
    BY REFERENCE  WIDTHS,
    BY VALUE      WIDTHS-NUM,
    BY VALUE      WIDTHS-SIZE
END CALL
```

Parameters:

SEARCHNAME is the name the search to used by the call.

WIDTHS is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

IDS-SERVER-VERSION-GET

Description:

Get the version information associated with the server.

Prototype:

```
CALL IDS-SERVER-VERSION-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SERVER-VERSION,
    BY VALUE      SERVER-VERSION-SIZE
END CALL
```

Parameters:

SERVER-VERSION is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

IDS-SESSION-CLOSE

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
CALL IDS-SESSION-CLOSE USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH
END CALL
```

Parameters:

none

Return Code:

negative for error, 0 for success

IDS-SESSION-OPEN

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
CALL IDS-SESSION-OPEN USING
      BY REFERENCE    RC,
      BY VALUE        SOCKH,
      BY REFERENCE    SESSION
END CALL
```

Parameters:

SESSION is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

IDS-SET-ENCODING

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the *Globalization* section of the OPERATIONS manual for further details and restrictions.

Prototype:

```
CALL IDS-SET-ENCODING USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY VALUE      ENCODING
END CALL
```

Parameters:

ENCODING 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

IDS-SET-TIMEOUT

Description:

Informs the Search Server of the timeout value in seconds. The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached).

Prototype:

```
CALL IDS-SET-TIMEOUT USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY VALUE      TIMEOUT
END CALL
```

Parameters:

TIMEOUT timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

IDS-SET-VPD-USER

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, *VPD* section for details about VPD.

Prototype:

```
CALL IDS-SET-VPD-USER USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  VPD-USER,
      BY REFERENCE  VPD-CTX
END CALL
```

Parameters:

VPD-USER Connection string of the actual user

VPD-CTX the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

IDS-SYSTEM-CLOSE

Description:

closes the system and frees any remaining resources.

Prototype:

```
CALL IDS-SYSTEM-CLOSE USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH
END CALL
```

Parameters:

none

Return Code:

negative for error, 0 for success

IDS-SYSTEM-IDTNAME-COUNT

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

```
CALL IDS-SYSTEM-IDTNAME-COUNT USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    IDTCOUNT
END CALL
```

Parameters:

IDTCOUNT is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

IDS-SYSTEM-IDTNAME-GET

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

```
CALL IDS-SYSTEM-IDTNAME-GET USING
    BY REFERENCE    RC,
    BY VALUE        SOCKH,
    BY REFERENCE    IDTNAMES,
    BY VALUE        IDTNAMES-NUM,
    BY VALUE        IDTNAMES-SIZE
END CALL
```

Parameters:

IDTNAMES is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

IDS-SYSTEM-NOTIFY

Description:

Notifies search server on a system.

Prototype:

```
CALL IDS-SYSTEM-NOTIFY USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  RULEBASE,
      BY REFERENCE  SYSNAME,
      BY REFERENCE  MESSAGE
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSNAME is the name of the system

MESSAGE is a message to be delivered

Return Code:

negative for error, 0 for success

IDS-SYSTEM-OPEN

Description:

opens a system.

Prototype:

```
CALL IDS-SYSTEM-OPEN USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  RULEBASE,
      BY REFERENCE  SYSTEM,
      BY REFERENCE  VERBOSITY,
      BY REFERENCE  OPTIONS
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSTEM The name of the system in the rulebase

VERBOSITY specifies the verbosity level. See the *Verbosity* section for details.

OPTIONS LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

IDS-SYSTEM-SEARCH-FINISH

Description:

Finishes the search and closes the system.

Prototype:

```
CALL IDS-SYSTEM-SEARCH-FINISH USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH
END CALL
```

Parameters:

none

Return Code:

negative for error, 0 for success

IDS-SYSTEM-SEARCH-START

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

```
CALL IDS-SYSTEM-SEARCH-START USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  RULEBASE,
      BY REFERENCE  SYSTEM,
      BY REFERENCE  VERBOSITY,
      BY REFERENCE  OPTIONS,
      BY REFERENCE  SEARCH,
      BY REFERENCE  PARAMETERS,
      BY REFERENCE  ANSWERSETNAME,
      BY REFERENCE  DATALEN,
      BY REFERENCE  RECS
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSTEM The name of the system in the rulebase

VERBOSITY specifies the verbosity level. See the *Verbosity* section for details.

OPTIONS consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1, KEYWORD2=VALUE2,

SEARCH is the name of the search in the system in the Rulebase that will be used.

PARAMETERS is the array which contains the field values to be used to construct the search.

ANSWERSETNAME is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

DATALEN will return the length of a record.

RECS count of records that matched the search criteria

Return Code:

negative for error, 0 for success

IDS-SYSTEM-SEARCHES-COUNT

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the SHOWALLSEARCHES keyword to the option string of ids_system_open.

Prototype:

```
CALL IDS-SYSTEM-SEARCHES-COUNT USING
      BY REFERENCE      RC,
      BY VALUE           SOCKH,
      BY REFERENCE      SEARCHCOUNT
END CALL
```

Parameters:

SEARCHCOUNT is the number of searches defined on the system

Return Code:

negative for error, 0 for success

IDS-SYSTEM-SEARCHES-GET

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the SHOWALLSEARCHES keyword to the option string of ids_system_open.

Prototype:

```
CALL IDS-SYSTEM-SEARCHES-GET USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  SEARCHES,
    BY VALUE      SEARCHES-NUM,
    BY VALUE      SEARCHES-SIZE
END CALL
```

Parameters:

SEARCHES is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

IDS-SYSTEMS-COUNT

Description:

the number of systems in the rulebase.

Prototype:

```
CALL IDS-SYSTEMS-COUNT USING
    BY REFERENCE  RC,
    BY VALUE      SOCKH,
    BY REFERENCE  RULEBASE,
    BY REFERENCE  SYSTEMSCOUNT
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSTEMSCOUNT the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

IDS-SYSTEMS-GET

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

```
CALL IDS-SYSTEMS-GET USING
      BY REFERENCE  RC,
      BY VALUE      SOCKH,
      BY REFERENCE  RULEBASE,
      BY REFERENCE  SYSTEMS,
      BY VALUE      SYSTEMS-NUM,
      BY VALUE      SYSTEMS-SIZE
END CALL
```

Parameters:

RULEBASE is the name of the rulebase.

SYSTEMS is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Calling from Java

Java Version

The Java components of IIR products were built with JDK V1.5. If you intend to build Java clients which make use of the published Java classes, then you should use JDK 1.5 or higher.

Overview

In Java the API methods are included in the jar file **idssecl.jar** (located in the %SSABIN% directory of the Client installation) as the class `ssa.ssase.ClieSock`. This is packaged as a Java Bean for use in IDEs. The methods throw exceptions of type `SSAAPIException` and `SSASocketException` that must be caught. If `SSASocketException` is caught, a fatal problem with the socket connection has been encountered. As a result, the communication channel has been closed and no further calls may be made without establishing a new connection.

Java does not require the `sockh` parameter for any of these functions.

Constants

Constants are declared as class constants, in uppercase using underscores, eg. `ClieSock.MSG_SIZE`.

Parameter types

Note: values in **BOLD** represent information that must be provided to the function.

Error Handling

If an error is detected during an API call, it is reported to the client program by throwing an exception. The class of the thrown exception varies depending upon the cause of the failure. Thus, the specific class of the thrown exception can be used to determine how the client program should proceed. There are three types of exception which may be thrown:-

SSAInterruptedException

This exception is thrown in the event that the process attempting to open a socket connection to the Search Server is interrupted before the connection can be established. The signature of the `ClieSock` constructor used determines when this attempt to open a socket occurs.

If a `ClieSock` constructor which accepts host & port parameters is used, then the connection is established during the constructor call. In this case, the `SSAInterruptedException` may be thrown by the `ClieSock` constructor call.

If a `ClieSock` constructor which does not accept host & port parameters is used, then the connection is established subsequent to the constructor call, by an explicit call to `ids_connect`. In this case, `ids_connect` may throw the `SSAInterruptedException` exception.

In either case, if this exception is thrown then the `ClieSock` object has not been initialized correctly and may not be used to make API calls to the Search Server.

SSAAPIException

This exception is thrown in the event that an error occurred on the Search Server while attempting to service an API call. As a result, the API call concerned did not complete successfully and the output parameters, if any, have not been set.

In response to an `SSAAPIException`, it is recommended that the client program should call the `getMessage()` & `getRc()` methods of the `SSAAPIException` object to build a description of the failure, for debugging purposes.

In addition, it is recommended that appropriate calls to the `ids_errors_get_all()` method of the `ClieSock` object be made to retrieve the error stack from the Search Server, again for debugging purposes.

Java code demonstrating how all these calls should be made can be found in the sample programs which accompany the distribution.

After an `SSAAPIException` has been caught, the connection to the Search Server remains intact and the `ClieSock` object may be used to make further API calls.

Note that the `SSAAPIException` class contains a `getFatal` method. This method is present for backward compatibility purposes only and need never be called.

public methods

`getRc`

This method returns the response code of the failed API call which caused the exception to be thrown

Prototype:

```
public int getRc()
```

Output:

The response code of the failed API call which caused the exception to be thrown

`getMessage`

This method returns the detail message string of this `SSAAPIException` instance

Prototype:

```
public String getMessage()
```

Output:

The detail message string of this `SSAAPIException` instance (which may be null)

SSASocketException

As the name implies, this exception is thrown in the event that a problem with the socket connection was encountered while attempting to communicate with the Search Server.

When an `SSASocketException` is thrown, the API call that was being made has failed and, in addition, the socket connection has been closed. No further API calls can be made using the current `ClieSock` object. Therefore, a new `ClieSock` object must be established before any further calls may be made.

Note that the `SSASocketException` class contains a `getFatal` method. This method is present for backward compatibility purposes only and need never be called.

public methods

`getMessage`

This method returns the detail message string of this `SSASocketException` instance

Prototype:

```
public String getMessage()
```

Output:

The detail message string of this `SSASocketException` instance (which may be null)

Deprecated APIs

In previous versions of IIR, error handling code of client programs caught an `SSAException` object and then called the methods `getRc()` and `getFatal()` of that object to determine the nature of a failure. As of version 9.0.00, the `SSAException` class is reserved for internal use only. In addition, that class's member methods, `getRc()` and `getFatal()`, have been deprecated. Client programs should not catch `SSAException`. Nor should they call either of the deprecated methods mentioned. Instead, user programs should follow the Error Handling guidelines above.

ClieSock Constructor

Description:

This class is used to create a `ClieSock` object. This object is then used to make the required API calls to the Search or Connection Server. There are several versions of the constructor.

The versions which accept host & port parameters establish a connection to the server during constructor execution. Therefore, the created `ClieSock` object is ready to perform API calls.

The versions which do not accept host & port parameters create the object but do not establish a connection to the server. If these constructors are used, it is necessary to call `ids_connect` to establish a connection to the server.

Prototype:

```
import ssa.ssase.ClieSock;

public ClieSock () throws SSASocketException;

public ClieSock (
    int      max_time_out          // long in
) throws SSASocketException, SSASocketException;

public ClieSock (
    String   hostname,             // String in
    int      port                  // long in
) throws SSASocketException, SSASocketException;

public ClieSock (
    String   hostname,             // String in
    int      port,                 // long in
    int      max_time_out          // long in
) throws SSASocketException, SSASocketException;
```

Parameters:

hostname is the hostname of the server to connect to

port is the port number of the server to connect to

max_time_out is the maximum time to attempt to establish a connection, expressed in milliseconds. The default value is 120000 ms.

ids_addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
using ClieSock;

public synchronized int ids_addr_get_cass_field (
    int      suggest_idx,          // Long in
    int      field_idx,           // Long in
    byte []  field_value,         // Block out
    int      field_value_size
) throws SSAPIException, SSASocketException;
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_cass_field_cnt (  
    int [] count // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_cass_field_info (  
    int suggest_idx, // Long in  
    int [] field_length, // LongArray out  
    int field_length_num  
) throws SSAPIException, SSASocketException;
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

ids_addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Prototype:

using ClieSock;

```
public synchronized int ids_addr_get_del_lines_ext (
    int          suggest_idx,    // Long in
    int          del_case,      // Long in
    byte []      del_line1,     // Block out
    int          del_line1_size,
    byte []      del_line2,     // Block out
    int          del_line2_size,
    byte []      del_line3,     // Block out
    int          del_line3_size,
    byte []      del_line4,     // Block out
    int          del_line4_size,
    byte []      del_line5,     // Block out
    int          del_line5_size,
    byte []      del_line6,     // Block out
    int          del_line6_size
) throws SSAPIException, SSASocketException;
```

Parameters:

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

ids_addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```
using ClieSock;

public synchronized int ids_addr_get_field (
    int          suggest_idx,    // Long in
    int          field_idx,     // Long in
    byte []      field_value,    // Block out
    int          field_value_size,
    int []       field_val_status, // Long out
    int []       field_val_mods  // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

ids_addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
using ClieSock;

public synchronized int ids_addr_get_field_count (
    int [] count // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

ids_addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Prototype:

```
using ClieSock ;

public synchronized int ids_addr_get_field_ext (
    int          suggest_idx ,    // Long in
    int          field_operation , // Long in
    String       field_name ,     // String in
    int          field_item_line , // Long in
    String       field_type ,     // String in
    byte []      field_value ,    // Block out
    int          field_value_size
) throws SSAPIException , SSASocketException ;
```

Parameters:

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

ids_addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_field_idx (  
    int          suggest_idx,    // Long in  
    int          field_idx,      // Long in  
    byte []      field_value,    // Block out  
    int          field_value_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

ids_addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_field_info_ext (  
    int          suggest_idx,    // Long in  
    int []       field_length,   // LongArray out  
    int          field_length_num,  
    byte []      addr_label_encoded, // Block out  
    int          addr_label_encoded_size,  
    String []    addr_label_charset, // String out  
    int          addr_label_charset_size,  
    int []       score           // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

ids_addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_field_len (  
    int []          max_len      // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_line_len (  
    int []          max_len      // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_get_option (  
    String      param,          // String in  
    String []   value,          // String out  
    int         value_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

ids_addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_info (  
    String      controls,        // String in  
    String []   value,          // String out  
    int         value_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

controls this field contains the request information. It must be specified in the form ITEM=[value].

value this field contains the requested information.

Return Code:

negative for error, 0 for success

ids_addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_init (  
    int max_memory // Long in  
) throws SSAPIException, SSASocketException;
```

Parameters:

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

ids_addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_parse (  
    int [] field_length, // LongArray out  
    int field_length_num  
) throws SSAPIException, SSASocketException;
```

Parameters:

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

ids_addr_preload_country

Description:

Use this function to preload country database

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_preload_country (  
    String          preload_type,    // String in  
    String          preload_country, // String in  
    String          val_mode        // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

preload_type Type of preload to perform

preload_country Country database to be preloaded

val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

ids_addr_set_attrib

Description:

Use this function to specify the character set of the data (for both input and output). The `default_country` parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_set_attrib (  
    String          char_set,        // String in  
    String          default_country // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

char_set The name of the character set used to encode the input and output.

default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

ids_addr_set_del_lines

Description:

Use this function to set delivery address line information

Prototype:

using ClieSock;

```
public synchronized int ids_addr_set_del_lines (
    byte []      del_line1 ,      // Block in
    byte []      del_line2 ,      // Block in
    byte []      del_line3 ,      // Block in
    byte []      del_line4 ,      // Block in
    byte []      del_line5 ,      // Block in
    byte []      del_line6       // Block in
) throws SSAPIException, SSASocketException;
```

Parameters:

del_line1 delivery address line 1 input string

del_line2 delivery address line 2 input string

del_line3 delivery address line 3 input string

del_line4 delivery address line 4 input string

del_line5 delivery address line 5 input string

del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

ids_addr_set_field_case

Description:

Use this function to set individual input fields case option

Prototype:

```
using ClieSock;

public synchronized int ids_addr_set_field_case (
    int          field_idx,      // Long in
    int          field_case     // Long in
) throws SSAPIException, SSASocketException;
```

Parameters:

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

ids_addr_set_field_ext

Description:

Use this function to set fields

Prototype:

```
using ClieSock;

public synchronized int ids_addr_set_field_ext (
    int          field_operation, // Long in
    String       field_name,     // String in
    int          field_item_line, // Long in
    String       field_type,     // String in
    byte []      field_value     // Block in
) throws SSAPIException, SSASocketException;
```

Parameters:

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

ids_addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_set_field_idx (  
    int          field_idx,      // Long in  
    byte []      field_value     // Block in  
) throws SSAPIException, SSASocketException;
```

Parameters:

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

ids_addr_set_field_name

Description:

Use this function to set individual input fields by name

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_set_field_name (  
    String       field_name,     // String in  
    byte []      field_value     // Block in  
) throws SSAPIException, SSASocketException;
```

Parameters:

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

ids_addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
using ClieSock;
```

```
public synchronized int ids_addr_set_lines (
    byte []    line_1 ,      // Block in
    byte []    line_2 ,      // Block in
    byte []    line_3 ,      // Block in
    byte []    line_4 ,      // Block in
    byte []    line_5 ,      // Block in
    byte []    line_6 ,      // Block in
    byte []    line_7 ,      // Block in
    byte []    line_8 ,      // Block in
    byte []    line_9 ,      // Block in
    byte []    line_10      // Block in
) throws SSAPIException , SSASocketException;
```

Parameters:

line_1 The first line of the address
line_2 The second line of the address
line_3 The third line of the address
line_4 The fourth line of the address
line_5 The fifth line of the address
line_6 The sixth line of the address
line_7 The seventh line of the address
line_8 The eighth line of the address
line_9 The ninth line of the address
line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

ids_addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
using ClieSock ;

public synchronized int ids_addr_set_option (
    String      param,          // String in
    String      value           // String in
) throws SSAPIException, SSASocketException;
```

Parameters:

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

ids_addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable `IDS Address Standardization Module` to be installed.

Prototype:

```
using ClieSock ;

public synchronized int ids_addr_std (
    String []    firm_name,      // String io
    int          firm_name_size ,
    String []    urbanization,   // String io
    int          urbanization_size ,
    String []    address_one,    // String io
    int          address_one_size ,
    String []    address_two,    // String io
    int          address_two_size ,
    String []    last_line,      // String io
    int          last_line_size
) throws SSAPIException, SSASocketException;
```

Parameters:

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

- 0 indicates an exact match to a valid address
- 1 indicates a no match (invalid address)
- 2 indicates a multi match (non-unique address), and
- < 0 indicates an error

ids_addr_validate

Description:

Use this function to validate an address

Prototype:

```
using ClieSock;  
  
public synchronized int ids_addr_validate (  
    int []      status ,           // Long out  
    int []      n_suggest         // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

status The status returned by the validation process

n_suggest The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

ids_disconnect

Description:

Releases resources allocated to a socket.

Prototype:

using ClieSock;

```
public synchronized int ids_disconnect () throws SSAPIException, SSASocketException;
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Prototype:

using ClieSock;

```
public synchronized int ids_error_get (
    String []    msg,           // String out
    int          msg_size
) throws SSAPIException, SSASocketException;
```

Parameters:

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

ids_errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for information on interpreting the Error Log.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_errors_get_all (  
    String []    msg,           // String out  
    int          msg_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

msg is an error message.

Return Code:

negative for error, 0 for success

ids_identify

Description:

Identify a session to the console

Prototype:

```
using ClieSock;  
  
public synchronized int ids_identify (  
    String          identification // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

ids_is_little_endian

Description:

Checks if the search server is running on a little endian platform

Prototype:

```
using ClieSock;

public synchronized int ids_is_little_endian (
    int []        endian_state    // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

ids_match_explain

Description:

Explains the match result given search and file records As match_explain_count does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

```
using ClieSock;

public synchronized int ids_match_explain (
    String        search,           // String in
    String        match_tolerance, // String in
    byte []       searchrec,        // Block in
    byte []       filerec,          // Block in
    byte [][]     info_array,       // BlockArray out
    int []        info_array_num,
    int []        info_array_size
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search which was performed.

match_tolerance specifies either Conservative, Typical or Loose to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

ids_match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

```
using ClieSock;  
  
public synchronized int ids_match_explain_count (  
    String          search,          // String in  
    int []          count            // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

ids_real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_real_time_async_get (  
    String          reference,        // String in  
    int             block,            // Long in  
    int []          cluster_action_count // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

ids_real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

```
using ClieSock;

public synchronized int ids_real_time_async_start (
    String      rulebase ,      // String in
    String      system ,       // String in
    String      IDT ,          // String in
    String      sequence_number , // String in
    String      operation ,    // String in
    byte []     cluster_record , // Block in
    int         source ,       // Long in
    String      multi_search ,  // String in
    int         input_id ,     // Long in
    String []   reference ,    // String out
    int         reference_size
) throws SSAPIException, SSASocketException;
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

ids_real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
using ClieSock ;

public synchronized int ids_real_time_flul_add (
    String          rule_type ,      // String in
    byte []         subject_rec_pk , // Block in
    String          relationship ,   // String in
    byte []         related_rec_pk   // Block in
) throws SSAPIException , SSASocketException ;
```

Parameters:

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

ids_real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Prototype:

```
using ClieSock;
```

```
public synchronized int ids_real_time_flul_close () throws SSAPIException, SSASocketException
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
using ClieSock;
```

```
public synchronized int ids_real_time_flul_delete (  
    int          rule_type_option, // Long in  
    byte []      record_pk        // Block in  
) throws SSAPIException, SSASocketException;
```

Parameters:

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

ids_real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_real_time_flul_find_rule (  
    byte []      idt_rec ,      // Block in  
    int          option         // Long in  
) throws SSAPIException, SSASocketException;
```

Parameters:

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_real_time_flul_get_rule (  
    byte []      idt_rec ,      // Block out  
    int          idt_rec_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_real_time_flul_init (  
    String      idt_name,        // String in  
    String      multi_search    // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

ids_real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_real_time_sync_get (  
    String      reference,        // String in  
    String []   cluster_action_type, // String out  
    int         cluster_action_type_size ,  
    String []   cluster_action_id, // String out  
    int         cluster_action_id_size ,  
    int []      cluster_action_number, // Long out  
    String []   cluster_action_new, // String out  
    int         cluster_action_new_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

ids_real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

using ClieSock;

```
public synchronized int ids_real_time_sync_start (
    String      rulebase ,           // String in
    String      system ,            // String in
    String      IDT ,               // String in
    String      sequence_number , // String in
    String      operation ,         // String in
    byte []     cluster_record ,    // Block in
    int []      cluster_action_count, // Long out
    String []   reference ,         // String out
    int         reference_size
) throws SSAPIException , SSASocketException;
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and postive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was lessa than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

ids_scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an Accept limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a SORT= parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

using ClieSock ;

```
public synchronized int ids_scores_get (
    String          searchname,      // String in
    int []          scores ,         // LongArray out
    int             scores_num
) throws SSAPIException , SSASocketException ;
```

Parameters:

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

ids_search_comment_get

Description:

Returns the user defined comment stored with the search.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_comment_get (  
    String          searchname,      // String in  
    String []       comment,         // String out  
    int             comment_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

ids_search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_dedupe_start (  
    String          search,           // String in  
    String          search_width,     // String in  
    String          match_tolerance,  // String in  
    byte [][]       parameters,       // BlockArray in  
    byte []         searchrec,        // Block io  
    int             searchrec_size,  
    String          AnswersetName,    // String in  
    int             flags,             // Long in  
    int []          recid,             // Long io  
    int []          recs              // Long io  
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the search that is to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the `AnswersetName` parameter with the Search-Record-Id (10 bytes) . The maximum `AnswersetName` length is 22 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the `recid` of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the `recid` of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

ids_search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
using ClieSock;

public synchronized int ids_search_fields_count (
    String      searchname,      // String in
    int []      fc               // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

ids_search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_fields_get (  
    String          searchname,      // String in  
    String []       fieldnames,      // StringArray out  
    int             fieldnames_num,  
    int             fieldnames_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

ids_search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the **SQL Filters** section of the DESIGNER MANUAL for details about SQL filters

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_filter (  
    String          search,          // String in  
    String          filter           // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

ids_search_finish

Description:

Release resources associated with `ids_search_start`.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_finish (  
    String search // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

ids_search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_get (  
    String searchname, // String in  
    byte [] searchreturn, // Block out  
    int searchreturn_size,  
    int [] score, // Long out  
    int [] sreps, // LongArray out  
    int sreps_num,  
    int [] freps, // LongArray out  
    int freps_num  
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and SSA-NAME3 v1 is used.

Prototype:

using ClieSock;

```
public synchronized int ids_search_get_complete (
    String          search,          // String in
    byte []         searchreturn,    // Block out
    int             searchreturn_size,
    int []          score,           // Long out
    byte []         info,           // Block out
    int             info_size
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4 \times (1 + 3 \times 100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score,

Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

```
using ClieSock;

public synchronized int ids_search_get_detail (
    String          search,          // String in
    byte []         file_rec,        // Block out
    int             file_rec_size,
    int []          score,           // Long out
    String []       decision,        // String out
    int             decision_size,
    int []          file_recid       // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Prototype:

```
using ClieSock ;

public synchronized int ids_search_IDT_get (
    String      searchname,    // String in
    String []   IDT,           // String out
    int         IDT_size
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

ids_search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

```
using ClieSock ;

public synchronized int ids_search_layout (
    String      search,        // String in
    String      viewType,      // String in
    String      func,          // String in
    String []   names,         // StringArray out
    int         names_num,
    int         names_size,
    int []      lengths,       // LongArray out
    int         lengths_num,
    int []      offsets,       // LongArray out
    int         offsets_num,
    int []      repeats,       // LongArray out
    int         repeats_num,
    String []   formats,       // StringArray out
    int         formats_num,
    int         formats_size
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters 2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the `FORMATS_SIZE` constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

ids_search_profile_count

Description:

Count the number of profile entries available

Prototype:

```
using ClieSock ;

public synchronized int ids_search_profile_count (
    int [] count // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

count Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_profile_get (  
    int []          ids ,           // LongArray out  
    int             ids_num ,  
    String []       names ,        // StringArray out  
    int             names_num ,  
    int             names_size ,  
    int []          times ,        // LongArray out  
    int             times_num  
) throws SSAPIException , SSASocketException ;
```

Parameters:

ids the numeric identifiers for each field, valid value are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

using ClieSock;

```
public synchronized int ids_search_start (
    String          search ,           // String in
    String          search_width ,     // String in
    String          match_tolerance ,  // String in
    byte [][]       parameters ,       // BlockArray in
    byte []         searchrec ,        // Block io
    int             searchrec_size ,
    String          AnswersetName ,    // String in
    int []          recs ,             // Long out
    byte [][]       records            // BlockArray in
) throws SSAPIException , SSASocketException;
```

Parameters:

search is the name of the search to be performed.

search_width specifies either *Narrow*, *Typical* or *Exhaustive* to nominate how many candidates should be selected. If left blank, the *Search_Level* in the Controls will be used.

match_tolerance specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the *Match_Level* in the Controls will be used.

parameters contains the field values used to construct a search record (*searchrec*). The order of the field values must correspond to the order of fields returned by *ids_search_fields_get*. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If *searchrec* is specified it will be used to search (provided no *parameters* are supplied). Alternatively if *parameters* are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an *AnswerSet*. *AnswersetName* is used to identify the Search results in the table. The maximum *AnswersetName* length is 32 characters. If you do not wish to store the search results in an *AnswerSet*, set *AnswersetName* to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the *Unique_Keys* option to remove duplicates, the resulting set may contain less than *recs* records. In this case, you must use the response code from *ids_search_get* to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding SEARCH_LIMIT and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

ids_search_start_via_parameters

Description:

Prototype:

using ClieSock;

```
public synchronized int ids_search_start_via_parameters (
    String          search,          // String in
    String          search_width,    // String in
    String          match_tolerance, // String in
    byte [][]       parameters,      // BlockArray in
    int []          datalen,         // Long out
    int []          recs             // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_start_via_record

Description:

Prototype:

```

using ClieSock;

public synchronized int ids_search_start_via_record (
    String          search,          // String in
    String          search_width,    // String in
    String          match_tolerance, // String in
    byte []         searchrec,       // Block in
    int []          datalen,         // Long out
    int []          recs            // Long out
) throws SSAPIException, SSASocketException;

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```

using ClieSock;

public synchronized int ids_search_tolerances_count (
    String          searchname,      // String in
    int []          count           // Long out
) throws SSAPIException, SSASocketException;

```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
using ClieSock;

public synchronized int ids_search_tolerances_get (
    String          searchname,      // String in
    String []       tolerances,      // StringArray out
    int             tolerances_num,
    int             tolerances_size
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Prototype:

```
using ClieSock;

public synchronized int ids_search_view_get (
    String      search ,           // String in
    String      viewType ,        // String in
    String []   viewName ,        // String out
    int         viewName_size ,
    int []      viewFieldCount ,  // Long out
    int []      viewRecLen       // Long out
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

ids_search_view_set

Description:

Sets a view as the active input or output view

Prototype:

```
using ClieSock;

public synchronized int ids_search_view_set (
    String      search ,           // String in
    String      viewType ,        // String in
    String      viewName          // String in
) throws SSAPIException, SSASocketException;
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

ids_search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_widths_count (  
    String          searchname,      // String in  
    int []          count            // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
using ClieSock;  
  
public synchronized int ids_search_widths_get (  
    String          searchname,      // String in  
    String []       widths,          // StringArray out  
    int             widths_num,  
    int             widths_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_server_version_get

Description:

Get the version information associated with the server.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_server_version_get (  
    String []      server_version, // String out  
    int            server_version_size  
) throws SSAPIException, SSASocketException;
```

Parameters:

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

ids_session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_session_close () throws SSAPIException, SSASocketException;
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server

from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
using ClieSock ;

public synchronized int ids_session_open (
    int [] session // Long io
) throws SSAPIException, SSASocketException;
```

Parameters:

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

ids_set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the *Globalization* section of the OPERATIONS manual for further details and restrictions.

Prototype:

```
using ClieSock ;

public synchronized int ids_set_encoding (
    int encoding // Long in
) throws SSAPIException, SSASocketException;
```

Parameters:

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

ids_set_timeout

Description:

Informs the Search Server of the timeout value in seconds. The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached).

Prototype:

```
using ClieSock;  
  
public synchronized int ids_set_timeout (  
    int timeout // Long in  
) throws SSAPIException, SSASocketException;
```

Parameters:

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

ids_set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, **VPD** section for details about VPD.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_set_vpd_user (  
    String vpd_user, // String in  
    String vpd_ctx // String in  
) throws SSAPIException, SSASocketException;
```

Parameters:

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

ids_system_close

Description:

closes the system and frees any remaining resources.

Prototype:

```
using ClieSock;  
  
public synchronized int ids_system_close () throws SSAPIException, SSASocketException;
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

```
using ClieSock;  
  
public synchronized int ids_system_idtname_count (  
    int [] idtcount // Long out  
) throws SSAPIException, SSASocketException;
```

Parameters:

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

ids_system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

```
using ClieSock;

public synchronized int ids_system_idtname_get (
    String []      idtnames,      // StringArray out
    int            idtnames_num,
    int            idtnames_size
) throws SSAPIException, SSASocketException;
```

Parameters:

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_system_notify

Description:

Notifies search server on a system.

Prototype:

```
using ClieSock;

public synchronized int ids_system_notify (
    String          rulebase,      // String in
    String          sysname,       // String in
    String          message        // String in
) throws SSAPIException, SSASocketException;
```

Parameters:

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

ids_system_open

Description:

opens a system.

Prototype:

```
using ClieSock ;

public synchronized int ids_system_open (
    String      rulebase ,      // String in
    String      system ,       // String in
    String      verbosity ,    // String in
    String      Options        // String in
) throws SSAPIException , SSASocketException ;
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

ids_system_search_finish

Description:

Finishes the search and closes the system.

Prototype:

```
using ClieSock ;

public synchronized int ids_system_search_finish () throws SSAPIException , SSASocketException
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

```
using ClieSock;

public synchronized int ids_system_search_start (
    String      rulebase ,      // String in
    String      system ,       // String in
    String      verbosity ,    // String in
    String      options ,      // String in
    String      search ,       // String in
    byte [][]   parameters ,    // BlockArray in
    String      AnswersetName , // String in
    int []      datalen ,       // Long out
    int []      recs           // Long out
) throws SSAPIException , SSASocketException;
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1, KEYWORD2=VALUE2,

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

datalen will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

ids_system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the SHOWALLSEARCHES keyword to the option string of `ids_system_open`.

Prototype:

```
using ClieSock ;

public synchronized int ids_system_searches_count (
    int []      searchcount    // Long out
) throws SSAPIException , SSASocketException ;
```

Parameters:

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

ids_system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
using ClieSock ;

public synchronized int ids_system_searches_get (
    String []      searches ,    // StringArray out
    int            searches_num ,
    int            searches_size
) throws SSAPIException , SSASocketException ;
```

Parameters:

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_systems_count

Description:

the number of systems in the rulebase.

Prototype:

```
using ClieSock ;

public synchronized int ids_systems_count (
    String      rulebase ,      // String in
    int []      systemscount    // Long out
) throws SSAPIException , SSASocketException ;
```

Parameters:

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

ids_systems_get

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

```
using ClieSock ;

public synchronized int ids_systems_get (
    String      rulebase ,      // String in
    String []   systems ,      // StringArray out
    int         systems_num ,
    int         systems_size
) throws SSAPIException , SSASocketException ;
```

Parameters:

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Calling from Perl

In Perl the API functions are in the module **ssajds.pm** (located in the **perl** directory of the Client and Developer Components installation).

The functions return a response code. A negative response code indicates a transport error, after which the communication channel is closed and no further API calls can be made without reconnecting.

Perl does not require the `sockh` value for any of these functions.

Constants - Object Oriented

Constants are declared as class variables, in uppercase using underscores, eg. `$ids->MSG_SIZE`.

Installation - Win32 client

To install this module type the following:

```
perl Makefile.PL
nmake
nmake test
nmake install
```

Installation - Unix client

To install this module type the following:

```
perl Makefile.PL
make
make test
make install
```

Parameter types

Note: values in **BOLD** represent information that must be provided to the function.

addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```

using ssa;

my $field_value = $ssa->addr_get_cass_field (
    $suggest_idx,    # Long in
    $field_idx       # Long in
);

```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```

using ssa;

my $count = $ssa->addr_get_cass_field_cnt ();

```

Parameters:

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
using ssa;  
  
my @field_length = $ssa->addr_get_cass_field_info (  
    $suggest_idx      # Long in  
);
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Prototype:

```
using ssa;  
  
my ($del_line1, $del_line2, $del_line3, $del_line4, $del_line5, $del_line6) = $ssa->addr_get_  
    $suggest_idx,      # Long in  
    $del_case          # Long in  
);
```

Parameters:

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```
using ssa;  
  
my ($field_value , $field_val_status , $field_val_mods) = $ssa->addr_get_field (  
    $suggest_idx ,    # Long in  
    $field_idx      # Long in  
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

```
using ssa;  
  
my $count = $ssa->addr_get_field_count ();
```

Parameters:

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Prototype:

```
using ssa;  
  
my $field_value = $ssa->addr_get_field_ext (  
    $suggest_idx,      # Long in  
    $field_operation, # Long in  
    $field_name,       # String in  
    $field_item_line, # Long in  
    $field_type        # String in  
);
```

Parameters:

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
using ssa;  
  
my $field_value = $ssa->addr_get_field_idx (  
    $suggest_idx,    # Long in  
    $field_idx       # Long in  
);
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
using ssa;  
  
my ($field_length, $addr_label_encoded, $addr_label_charset, $score) = $ssa->addr_get_field_info_ext(  
    $suggest_idx    # Long in  
);
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

```
using ssa;  
  
my $max_len = $ssa->addr_get_field_len ();
```

Parameters:

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

```
using ssa;  
  
my $max_len = $ssa->addr_get_line_len ();
```

Parameters:

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
using ssa;  
  
my $value = $ssa->addr_get_option (  
    $param          # String in  
);
```

Parameters:

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

```
using ssa;  
  
my $value = $ssa->addr_info (  
    $controls        # String in  
);
```

Parameters:

controls this field contains the request information. It must be specified in the form `ITEM=[value]`.

value this field contains the requested information.

Return Code:

negative for error, 0 for success

addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

```
using ssa;  
  
$ssa->addr_init (  
    $max_memory      # Long in  
);
```

Parameters:

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

```
using ssa;  
  
my @field_length = $ssa->addr_parse ();
```

Parameters:

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

addr_preload_country

Description:

Use this function to preload country database

Prototype:

```
using ssa;

$ssa->addr_preload_country (
    $preload_type, # String in
    $preload_country, # String in
    $val_mode # String in
);
```

Parameters:

preload_type Type of preload to perform
preload_country Country database to be preloaded
val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

addr_set_attrb

Description:

Use this function to specify the character set of the data (for both input and output). The `default_country` parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

```
using ssa;

$ssa->addr_set_attrb (
    $char_set, # String in
    $default_country # String in
);
```

Parameters:

char_set The name of the character set used to encode the input and output.
default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

addr_set_del_lines

Description:

Use this function to set delivery address line information

Prototype:

```
using ssa;

$ssa->addr_set_del_lines (
    $del_line1 ,      # Block in
    $del_line1_size ,
    $del_line2 ,      # Block in
    $del_line2_size ,
    $del_line3 ,      # Block in
    $del_line3_size ,
    $del_line4 ,      # Block in
    $del_line4_size ,
    $del_line5 ,      # Block in
    $del_line5_size ,
    $del_line6 ,      # Block in
    $del_line6_size
);
```

Parameters:

del_line1 delivery address line 1 input string

del_line2 delivery address line 2 input string

del_line3 delivery address line 3 input string

del_line4 delivery address line 4 input string

del_line5 delivery address line 5 input string

del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

addr_set_field_case

Description:

Use this function to set individual input fields case option

Prototype:

```
using ssa;

$ssa->addr_set_field_case (
    $field_idx ,      # Long in
    $field_case       # Long in
);
```

Parameters:

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

addr_set_field_ext

Description:

Use this function to set fields

Prototype:

```
using ssa;  
  
$ssa->addr_set_field_ext (  
    $field_operation, # Long in  
    $field_name,      # String in  
    $field_item_line, # Long in  
    $field_type,      # String in  
    $field_value,     # Block in  
    $field_value_size  
);
```

Parameters:

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
using ssa;

$ssa->addr_set_field_idx (
    $field_idx,      # Long in
    $field_value,    # Block in
    $field_value_size
);
```

Parameters:

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

addr_set_field_name

Description:

Use this function to set individual input fields by name

Prototype:

```
using ssa;

$ssa->addr_set_field_name (
    $field_name,      # String in
    $field_value,     # Block in
    $field_value_size
);
```

Parameters:

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

```
using ssa;

$ssa->addr_set_lines (
    $line_1 ,      # Block in
    $line_1_size ,
    $line_2 ,      # Block in
    $line_2_size ,
    $line_3 ,      # Block in
    $line_3_size ,
    $line_4 ,      # Block in
    $line_4_size ,
    $line_5 ,      # Block in
    $line_5_size ,
    $line_6 ,      # Block in
    $line_6_size ,
    $line_7 ,      # Block in
    $line_7_size ,
    $line_8 ,      # Block in
    $line_8_size ,
    $line_9 ,      # Block in
    $line_9_size ,
    $line_10 ,     # Block in
    $line_10_size
);
```

Parameters:

line_1 The first line of the address
line_2 The second line of the address
line_3 The third line of the address
line_4 The fourth line of the address
line_5 The fifth line of the address
line_6 The sixth line of the address
line_7 The seventh line of the address
line_8 The eighth line of the address
line_9 The ninth line of the address
line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

```
using ssa;

$ssa->addr_set_option (
    $param,          # String in
    $value           # String in
);
```

Parameters:

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable `IDS Address Standardization Module` to be installed.

Prototype:

```
using ssa;

$ssa->addr_std (
    \ $firm_name,      # String io
    $firm_name_size,
    \ $urbanization,   # String io
    $urbanization_size,
    \ $address_one,    # String io
    $address_one_size,
    \ $address_two,    # String io
    $address_two_size,
    \ $last_line,      # String io
    $last_line_size
);
```

Parameters:

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when **address_one** is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

- 0 indicates an exact match to a valid address
- 1 indicates a no match (invalid address)
- 2 indicates a multi match (non-unique address), and
- < 0 indicates an error

addr_validate

Description:

Use this function to validate an address

Prototype:

```
using ssa;  
  
my ($status $n_suggest) = $ssa->addr_validate ();
```

Parameters:

- status** The status returned by the validation process
- n_suggest** The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

disconnect

Description:

Releases resources allocated to a socket.

Prototype:

```
using ssa;  
  
$ssa->disconnect ();
```

Parameters:

- none

Return Code:

negative for error, 0 for success

error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Prototype:

```
using ssa;  
  
my $msg = $ssa->error_get ();
```

Parameters:

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for information on interpreting the Error Log.

Prototype:

```
using ssa;  
  
my $msg = $ssa->errors_get_all ();
```

Parameters:

msg is an error message.

Return Code:

negative for error, 0 for success

identify

Description:

Identify a session to the console

Prototype:

```
using ssa;  
  
$ssa->identify (  
    $identification # String in  
);
```

Parameters:

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

is_little_endian

Description:

Checks if the search server is running on a little endian platform

Prototype:

```
using ssa;  
  
my $endian_state = $ssa->is_little_endian ();
```

Parameters:

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

match_explain

Description:

Explains the match result given search and file records As `match_explain_count` does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

```
using ssa;  
  
my @info_array = $ssa->match_explain (  
    $search,          # String in  
    $match_tolerance, # String in  
    $searchrec,       # Block in  
    $searchrec_size,  #  
    $filerec,         # Block in  
    $filerec_size  
);
```

Parameters:

search is the name of the Search which was performed.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

```
using ssa;  
  
my $count = $ssa->match_explain_count (  
    $search          # String in  
);
```

Parameters:

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

```
using ssa;  
  
my $cluster_action_count = $ssa->real_time_async_get (  
    $reference,      # String in  
    $block          # Long in  
);
```

Parameters:

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

```
using ssa;

my $reference = $ssa->real_time_async_start (
    $rulebase,      # String in
    $system,        # String in
    $IDT,           # String in
    $sequence_number, # String in
    $operation,     # String in
    $cluster_record, # Block in
    $cluster_record_size,
    $source,        # Long in
    $multi_search,  # String in
    $input_id       # Long in
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with `WARN_DUPLICATE_PK` sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
using ssa;

$ssa->real_time_flul_add (
    $rule_type,      # String in
    $subject_rec_pk, # Block in
    $subject_rec_pk_size,
    $relationship,   # String in
    $related_rec_pk, # Block in
    $related_rec_pk_size
);
```

Parameters:

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Prototype:

```
using ssa;

$ssa->real_time_flul_close ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

```
using ssa ;

$ssa->real_time_flul_delete (
    $rule_type_option ,# Long in
    $record_pk ,      # Block in
    $record_pk_size
);
```

Parameters:

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
using ssa ;

$ssa->real_time_flul_find_rule (
    $idt_rec ,      # Block in
    $idt_rec_size ,
    $option         # Long in
);
```

Parameters:

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
using ssa;  
  
my $idt_rec = $ssa->real_time_flul_get_rule ();
```

Parameters:

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

```
using ssa;  
  
$ssa->real_time_flul_init (  
    $idt_name,      # String in  
    $multi_search  # String in  
);
```

Parameters:

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

```
using ssa;
```

```
my ($cluster_action_type, $cluster_action_id, $cluster_action_number, $cluster_action_new) = $  
    $reference           # String in  
);
```

Parameters:

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

```
using ssa;

my ($cluster_action_count, $reference) = $ssa->real_time_sync_start (
    $rulebase,          # String in
    $system,            # String in
    $IDT,               # String in
    $sequence_number,   # String in
    $operation,         # String in
    $cluster_record,    # Block in
    $cluster_record_size
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and postive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was lessa than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an `Accept` limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a `SORT=` parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

```
using ssa;  
  
my @scores = $ssa->scores_get (  
    $searchname      # String in  
);
```

Parameters:

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

search_comment_get

Description:

Returns the user defined comment stored with the search.

Prototype:

```
using ssa;  
  
my $comment = $ssa->search_comment_get (  
    $searchname      # String in  
);
```

Parameters:

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

```
using ssa;

$ssa->search_dedupe_start (
    $search,          # String in
    $search_width,    # String in
    $match_tolerance, # String in
    \@parameters,     # BlockArray in
    \$searchrec,       # Block io
    $searchrec_size,
    $AnswersetName,    # String in
    $flags,             # Long in
    \$recid,           # Long io
    \$recs              # Long io
);
```

Parameters:

search is the name of the search that is to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the `AnswersetName` parameter with the Search-Record-Id (10 bytes) . The maximum `AnswersetName` length is 22 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the `recid` of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the `recid` of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
using ssa ;  
  
my $fc = $ssa->search_fields_count (  
    $searchname      # String in  
);
```

Parameters:

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Prototype:

```
using ssa ;  
  
my @fieldnames = $ssa->search_fields_get (  
    $searchname      # String in  
);
```

Parameters:

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the *SQL Filters* section of the DESIGNER MANUAL for details about SQL filters

Prototype:

```
using ssa ;

$ssa->search_filter (
    $search ,      # String in
    $filter        # String in
);
```

Parameters:

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

search_finish

Description:

Release resources associated with `ids_search_start`.

Prototype:

```
using ssa ;

$ssa->search_finish (
    $search        # String in
);
```

Parameters:

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

```
using ssa;  
  
my ($searchreturn, $score, $sreps, $freps) = $ssa->search_get (  
    $searchname      # String in  
);
```

Parameters:

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and `SSA-NAME3 v1` is used.

Prototype:

```
using ssa;

my ($searchreturn, $score, $info) = $ssa->search_get_complete (
    $search          # String in
);
```

Parameters:

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4 \times (1 + 3 \times 100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

```
using ssa;

my ($file_rec, $score, $decision, $file_recid) = $ssa->search_get_detail (
    $search          # String in
);
```

Parameters:

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Prototype:

```
using ssa;  
  
my $IDT = $ssa->search_IDT_get (  
    $searchname      # String in  
);
```

Parameters:

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

```
using ssa;  
  
my ($names, $lengths, $offsets, $repeats, $formats) = $ssa->search_layout (  
    $search,      # String in  
    $viewType,    # String in  
    $func         # String in  
);
```

Parameters:

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters 2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the `FORMATS_SIZE` constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

search_profile_count

Description:

Count the number of profile entries available

Prototype:

```
using ssa;  
  
my $count = $ssa->search_profile_count ();
```

Parameters:

count Returns the number of elements

Return Code:

negative for error, 0 for success

search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

```
using ssa;  
  
my ($ids $names, $times) = $ssa->search_profile_get ();
```

Parameters:

ids the numeric identifiers for each field, valid value are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

```
using ssa;  
  
my $recs = $ssa->search_start (  
    $search,          # String in  
    $search_width,    # String in
```

```

    $match_tolerance, # String in
    \@parameters,    # BlockArray in
    \$searchrec,      # Block io
    $searchrec_size,
    $AnswersetName,   # String in
    \@records         # BlockArray in
);

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an `AnswerSet`. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an `AnswerSet`, set `AnswersetName` to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding `SEARCH_LIMIT` and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

search_start_via_parameters

Description:

Prototype:

```

using ssa;

my ($datalen, $recs) = $ssa->search_start_via_parameters (
    $search,          # String in

```

```

    $search_width,    # String in
    $match_tolerance, # String in
    \@parameters     # BlockArray in
);

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

search_start_via_record

Description:

Prototype:

```

using ssa;

my ($datalen, $recs) = $ssa->search_start_via_record (
    $search,          # String in
    $search_width,    # String in
    $match_tolerance, # String in
    $searchrec,       # Block in
    $searchrec_size
);

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If **searchrec** is specified it will be used to search (provided no **parameters** are supplied). Alternatively if **parameters** are specified then the search will be on a record constructed on those parameters and returned to the user.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than **recs** records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
using ssa;  
  
my $count = $ssa->search_tolerances_count (  
    $searchname      # String in  
);
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

```
using ssa;  
  
my @tolerances = $ssa->search_tolerances_get (  
    $searchname      # String in  
);
```

Parameters:

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Prototype:

```
using ssa;  
  
my ($viewName, $viewFieldCount, $viewRecLen) = $ssa->search_view_get (  
    $search,      # String in  
    $viewType     # String in  
);
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

search_view_set

Description:

Sets a view as the active input or output view

Prototype:

```
using ssa;  
  
$ssa->search_view_set (  
    $search ,      # String in  
    $viewType ,   # String in  
    $viewName     # String in  
);
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
using ssa;  
  
my $count = $ssa->search_widths_count (  
    $searchname    # String in  
);
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
using ssa;  
  
my @widths = $ssa->search_widths_get (  
    $searchname      # String in  
);
```

Parameters:

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

server_version_get

Description:

Get the version information associated with the server.

Prototype:

```
using ssa;  
  
my $server_version = $ssa->server_version_get ();
```

Parameters:

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
using ssa;  
$ssa->session_close ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
using ssa;  
$ssa->session_open (  
    \ $session          # Long io  
);
```

Parameters:

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the **Globalization** section of the OPERATIONS manual for further details and restrictions.

Prototype:

```
using ssa;  
  
$ssa->set_encoding (   
    $encoding          # Long in  
);
```

Parameters:

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

set_timeout

Description:

Informs the Search Server of the timeout value in seconds The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached)

Prototype:

```
using ssa;  
  
$ssa->set_timeout (   
    $timeout          # Long in  
);
```

Parameters:

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, *VPD* section for details about VPD.

Prototype:

```
using ssa;  
  
$ssa->set_vpd_user (  
    $vpd_user ,      # String in  
    $vpd_ctx         # String in  
);
```

Parameters:

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

system_close

Description:

closes the system and frees any remaining resources.

Prototype:

```
using ssa;  
  
$ssa->system_close ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

```
using ssa;  
  
my $idtcount = $ssa->system_idtname_count ();
```

Parameters:

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

```
using ssa ;  
  
my @idtnames = $ssa->system_idtname_get ( );
```

Parameters:

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

system_notify

Description:

Notifies search server on a system.

Prototype:

```
using ssa ;  
  
$ssa->system_notify (   
    $rulebase ,      # String in  
    $sysname ,       # String in  
    $message         # String in  
);
```

Parameters:

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

system_open

Description:

opens a system.

Prototype:

```
using ssa ;

$ssa->system_open (
    $rulebase ,      # String in
    $system ,        # String in
    $verbosity ,     # String in
    $Options         # String in
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

system_search_finish

Description:

Finishes the search and closes the system.

Prototype:

```
using ssa;  
  
$ssa->system_search_finish ();
```

Parameters:

none

Return Code:

negative for error, 0 for success

system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

```
using ssa;  
  
my ($datalen, $recs) = $ssa->system_search_start (  
    $rulebase,      # String in  
    $system,        # String in  
    $verbosity,     # String in  
    $options,       # String in  
    $search,        # String in  
    \@parameters,  # BlockArray in  
    $AnswersetName  # String in  
);
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form `KEYWORD1=VALUE1,KEYWORD2=VALUE2,`

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

datalen will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
using ssa;  
  
my $searchcount = $ssa->system_searches_count ();
```

Parameters:

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Prototype:

```
using ssa;  
  
my @searches = $ssa->system_searches_get ();
```

Parameters:

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

systems_count

Description:

the number of systems in the rulebase.

Prototype:

```
using ssa;  
  
my $systemscount = $ssa->systems_count (  
    $rulebase      # String in  
);
```

Parameters:

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

systems_get

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

```
using ssa;  
  
my @systems = $ssa->systems_get (  
    $rulebase      # String in  
);
```

Parameters:

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Calling from Visual Basic .NET

For Visual Basic .NET the API functions are made available in the shareable assembly **ssasecs.dll**. This contains the classes `ids`, which contains the API, and `SSAException`, which is based on `ApplicationException`. The `ids` methods return an exception of class `SSAException` that must be caught. If the `bool` method `SSAException.IsFatal()` returns `true` the error resulted in the communication channel being closed and no further calls can be made without reconnecting.

Visual Basic .NET does not require the `sockh` parameter for any of these functions.

Constants

Constants are public member constants of the `ids` class, in uppercase using underscores, eg. `ids.MSG_SIZE`.

Installation - Win32 client

The shareable assembly can be installed in the global assembly cache with Microsoft's **gacutil** utility:

```
gacutil /i %SSABIN%\ssasecs.dll
```

You can then create applications that use it with:

```
vbc /reference:%SSABIN%\ssasecs.dll myprog.vb
```

Alternatively, you can use **ssasecs.dll** in a Visual Studios.NET IDE project/solution.

To add a reference to the **ssasecs.dll**:

- ◆ Right-click on the project/solution name in the "Solution Explorer" pane of the IDE. Select "Add Reference...".
- ◆ Make sure that the ".NET" tab is the active tab. Use the "Browse" button to find **ssasecs.dll**.
- ◆ Highlight the dll in the Explorer window (*contrary to expectations, nothing you do will add the name of the dll to the "File Name:" textbox, so don't worry if it's not there*) and click "Open".
- ◆ This adds the **ssasecs.dll** to the Selected Components pane of the Add Reference dialog. Click "OK".

To add the "ssa" namespace to the project/solution:

- ◆ Right-Click on the project/solution name in the "Solution Explorer" pane of the IDE. Select "Properties".
- ◆ Click on "Imports" under the "Common Properties" folder.
- ◆ In the "Namespace:" textbox type `ssa` and click "Add Import".
- ◆ Click OK to exit.

The second step eliminates the need to have the following in any module:

```
Imports ssa
```

Otherwise, you can skip this step and add the `Imports` statement as above.

Constructor

Description:

Used to create a `ssa.ids` object. This object is then used to make the required API calls to the Search or Connection Server.

Prototype:

```
Imports ssa;  
  
Public Class ids  
    Dim hostname As String  
    Dim port As Integer  
End Class
```

Parameters:

hostname is the name or IP address of the computer where the IIR Server is running

port the port number of the IIR Server for which a socket connection is to be established

addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

```
Imports ssa  
  
Public Function addr_get_cass_field ( _  
    ByRef suggest_idx As Integer, _' Long in  
    ByRef field_idx As Integer _' Long in  
) As Byte ()
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Prototype:

Imports ssa

Public Function addr_get_cass_field_cnt () **As Integer**

Parameters:

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

Imports ssa

Public Function addr_get_cass_field_info (_
 ByRef suggest_idx **As Integer** _' Long in
) **As Integer** ()

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Prototype:

Imports ssa

```
Public Function addr_get_del_lines_ext ( _  
    ByRef suggest_idx As Integer, _' Long in  
    ByRef del_case As Integer _' Long in  
) As addr_get_del_lines_ext_struct
```

Parameters:

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Prototype:

```
Imports ssa
```

```
Public Function addr_get_field ( _  
    ByRef suggest_idx As Integer, _' Long in  
    ByRef field_idx As Integer _' Long in  
) As addr_get_field_struct
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the field_length array for the `ids_addr_parse` API.

Prototype:

```
Imports ssa
```

```
Public Function addr_get_field_count () As Integer
```

Parameters:

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Prototype:

Imports ssa

```
Public Function addr_get_field_ext ( _  
    ByRef suggest_idx As Integer, _' Long in  
    ByRef field_operation As Integer, _' Long in  
    ByRef field_name As String, _' String in  
    ByRef field_item_line As Integer, _' Long in  
    ByRef field_type As String _' String in  
) As Byte ()
```

Parameters:

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Prototype:

Imports ssa

```
Public Function addr_get_field_idx ( _  
    ByRef suggest_idx As Integer, _' Long in  
    ByRef field_idx As Integer _' Long in  
) As Byte ()
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

Imports ssa

```
Public Function addr_get_field_info_ext ( _  
    ByRef suggest_idx As Integer _' Long in  
) As addr_get_field_info_ext_struct
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Prototype:

Imports ssa

```
Public Function addr_get_field_len () As Integer
```

Parameters:

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Prototype:

Imports ssa

Public Function addr_get_line_len () **As Integer**

Parameters:

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

Imports ssa

Public Function addr_get_option (_
 ByRef param **As String** _' *String in*
) **As String**

Parameters:

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Prototype:

Imports ssa

```
Public Function addr_info ( _  
    ByRef controls As String _' String in  
) As String
```

Parameters:

controls this field contains the request information. It must be specified in the form ITEM=[value].

value this field contains the requested information.

Return Code:

negative for error, 0 for success

addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Prototype:

Imports ssa

```
Public Sub addr_init ( _  
    ByRef max_memory As Integer _' Long in  
)
```

Parameters:

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Prototype:

Imports ssa

Public Function addr_parse () **As Integer** ()

Parameters:

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

addr_preload_country

Description:

Use this function to preload country database

Prototype:

Imports ssa

```
Public Sub addr_preload_country ( _  
    ByRef preload_type As String, _' String in  
    ByRef preload_country As String, _' String in  
    ByRef val_mode As String _' String in  
)
```

Parameters:

preload_type Type of preload to perform

preload_country Country database to be preloaded

val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

addr_set_attrib

Description:

Use this function to specify the character set of the data (for both input and output). The `default_country` parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Prototype:

Imports ssa

```
Public Sub addr_set_attrib ( _  
    ByRef char_set As String, _' String in  
    ByRef default_country As String _' String in  
)
```

Parameters:

char_set The name of the character set used to encode the input and output.

default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

addr_set_del_lines

Description:

Use this function to set delivery address line information

Prototype:

Imports ssa

```
Public Sub addr_set_del_lines ( _  
    ByRef del_line1() As Byte, _' Block in  
    ByRef del_line2() As Byte, _' Block in  
    ByRef del_line3() As Byte, _' Block in  
    ByRef del_line4() As Byte, _' Block in  
    ByRef del_line5() As Byte, _' Block in  
    ByRef del_line6() As Byte _' Block in  
)
```

Parameters:

del_line1 delivery address line 1 input string

del_line2 delivery address line 2 input string

del_line3 delivery address line 3 input string

del_line4 delivery address line 4 input string

del_line5 delivery address line 5 input string

del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

addr_set_field_case

Description:

Use this function to set individual input fields case option

Prototype:

Imports ssa

```
Public Sub addr_set_field_case ( _  
    ByRef field_idx As Integer, _' Long in  
    ByRef field_case As Integer _' Long in  
)
```

Parameters:

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

addr_set_field_ext

Description:

Use this function to set fields

Prototype:

Imports ssa

```
Public Sub addr_set_field_ext ( _  
    ByRef field_operation As Integer, _' Long in  
    ByRef field_name As String, _' String in  
    ByRef field_item_line As Integer, _' Long in  
    ByRef field_type As String, _' String in  
    ByRef field_value() As Byte _' Block in  
)
```

Parameters:

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

Imports ssa

```
Public Sub addr_set_field_idx ( _  
    ByRef field_idx As Integer, _' Long in  
    ByRef field_value() As Byte _' Block in  
)
```

Parameters:

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

addr_set_field_name

Description:

Use this function to set individual input fields by name

Prototype:

Imports ssa

```
Public Sub addr_set_field_name ( _  
    ByRef field_name As String, _' String in  
    ByRef field_value() As Byte _' Block in  
)
```

Parameters:

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Prototype:

Imports ssa

```
Public Sub addr_set_lines ( _  
    ByRef line_1() As Byte, _' Block in  
    ByRef line_2() As Byte, _' Block in  
    ByRef line_3() As Byte, _' Block in  
    ByRef line_4() As Byte, _' Block in  
    ByRef line_5() As Byte, _' Block in  
    ByRef line_6() As Byte, _' Block in  
    ByRef line_7() As Byte, _' Block in  
    ByRef line_8() As Byte, _' Block in  
    ByRef line_9() As Byte, _' Block in  
    ByRef line_10() As Byte _' Block in  
)
```

Parameters:

line_1 The first line of the address

line_2 The second line of the address
line_3 The third line of the address
line_4 The fourth line of the address
line_5 The fifth line of the address
line_6 The sixth line of the address
line_7 The seventh line of the address
line_8 The eighth line of the address
line_9 The ninth line of the address
line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Prototype:

Imports ssa

```
Public Sub addr_set_option ( _  
    ByRef param As String, _' String in  
    ByRef value As String _' String in  
)
```

Parameters:

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable *IDS Address Standardization Module* to be installed.

Prototype:

Imports ssa

```
Public Sub addr_std ( _  
    ByVal firm_name As String, _' String io  
    ByRef firm_name_size As Integer, _  
    ByVal urbanization As String, _' String io  
    ByRef urbanization_size As Integer, _  
    ByVal address_one As String, _' String io  
    ByRef address_one_size As Integer, _  
    ByVal address_two As String, _' String io  
    ByRef address_two_size As Integer, _  
    ByVal last_line As String, _' String io  
    ByRef last_line_size As Integer _  
)
```

Parameters:

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

0 indicates an exact match to a valid address

1 indicates a no match (invalid address)

2 indicates a multi match (non-unique address), and

< 0 indicates an error

addr_validate

Description:

Use this function to validate an address

Prototype:

Imports ssa

```
Public Function addr_validate () As addr_validate_struct
```

Parameters:

status The status returned by the validation process

n_suggest The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

disconnect

Description:

Releases resources allocated to a socket.

Prototype:

Imports ssa

Public Sub disconnect ()

Parameters:

none

Return Code:

negative for error, 0 for success

error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Prototype:

Imports ssa

Public Function error_get () **As String**

Parameters:

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for information on interpreting the Error Log.

Prototype:

Imports ssa

Public Function errors_get_all () **As String**

Parameters:

msg is an error message.

Return Code:

negative for error, 0 for success

identify

Description:

Identify a session to the console

Prototype:

Imports ssa

Public Sub identify (_
 ByRef identification **As String** _' *String in*
)

Parameters:

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

is_little_endian

Description:

Checks if the search server is running on a little endian platform

Prototype:

Imports ssa

Public Function is_little_endian () **As Integer**

Parameters:

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

match_explain

Description:

Explains the match result given search and file records As match_explain_count does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Prototype:

Imports ssa

```
Public Function match_explain ( _  
    ByRef search As String, _' String in  
    ByRef match_tolerance As String, _' String in  
    ByRef searchrec() As Byte, _' Block in  
    ByRef filerec() As Byte _' Block in  
) As Byte ()()
```

Parameters:

search is the name of the Search which was performed.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Prototype:

Imports ssa

```
Public Function match_explain_count ( _  
    ByRef search As String _' String in  
) As Integer
```

Parameters:

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Prototype:

Imports ssa

```
Public Function real_time_async_get ( _  
    ByRef reference As String, _' String in  
    ByRef block As Integer _' Long in  
) As Integer
```

Parameters:

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Prototype:

Imports ssa

```
Public Function real_time_async_start ( _  
    ByRef rulebase As String, _' String in  
    ByRef system As String, _' String in  
    ByRef IDT As String, _' String in  
    ByRef sequence_number As String, _' String in  
    ByRef operation As String, _' String in  
    ByRef cluster_record() As Byte, _' Block in  
    ByRef source As Integer, _' Long in  
    ByRef multi_search As String, _' String in  
    ByRef input_id As Integer _' Long in  
) As String
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Prototype:

Imports ssa

```
Public Sub real_time_flul_add ( _  
    ByRef rule_type As String, _' String in  
    ByRef subject_rec_pk() As Byte, _' Block in  
    ByRef relationship As String, _' String in  
    ByRef related_rec_pk() As Byte _' Block in  
)
```

Parameters:

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Prototype:

Imports ssa

Public Sub real_time_flul_close ()

Parameters:

none

Return Code:

negative for error, 0 for success

real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by `ids_real_time_flul_init`.

Prototype:

Imports ssa

```
Public Sub real_time_flul_delete ( _  
    ByRef rule_type_option As Integer, _' Long in  
    ByRef record_pk() As Byte _' Block in  
)
```

Parameters:

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after `ids_real_time_flul_init` API.

Prototype:

```
Imports ssa

Public Sub real_time_flul_find_rule ( _
    ByRef idt_rec() As Byte, _' Block in
    ByRef option As Integer _' Long in
)
```

Parameters:

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Prototype:

```
Imports ssa

Public Function real_time_flul_get_rule () As Byte ()
```

Parameters:

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Prototype:

Imports ssa

```
Public Sub real_time_flul_init ( _  
    ByRef idt_name As String, _' String in  
    ByRef multi_search As String _' String in  
)
```

Parameters:

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Prototype:

Imports ssa

```
Public Function real_time_sync_get ( _  
    ByRef reference As String _' String in  
) As real_time_sync_get_struct
```

Parameters:

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Prototype:

Imports ssa

```
Public Function real_time_sync_start ( _  
    ByRef rulebase As String, _' String in  
    ByRef system As String, _' String in  
    ByRef IDT As String, _' String in  
    ByRef sequence_number As String, _' String in  
    ByRef operation As String, _' String in  
    ByRef cluster_record() As Byte _' Block in  
) As real_time_sync_start_struct
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and positive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with WARN_DUPLICATE_PK sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was lessa than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an Accept limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a `SORT=` parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Prototype:

Imports ssa

```
Public Function scores_get ( _  
                        ByRef searchname As String _' String in  
) As Integer ()
```

Parameters:

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

search_comment_get

Description:

Returns the user defined comment stored with the search.

Prototype:

Imports ssa

```
Public Function search_comment_get ( _  
    ByRef searchname As String _' String in  
) As String
```

Parameters:

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Prototype:

Imports ssa

```
Public Sub search_dedupe_start ( _  
    ByRef search As String, _' String in  
    ByRef search_width As String, _' String in  
    ByRef match_tolerance As String, _' String in  
    ByRef parameters() As Byte, _' BlockArray in  
    ByVal searchrec() As Byte, _' Block io  
    ByRef searchrec_size As Integer, _  
    ByRef AnswersetName As String, _' String in  
    ByRef flags As Integer, _' Long in  
    ByVal recid As Integer, _' Long io  
    ByVal recs As Integer _' Long io  
)
```

Parameters:

search is the name of the search that is to be performed.

search_width specifies either *Narrow*, *Typical* or *Exhaustive* to nominate how many candidates should be selected.

match_tolerance specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used to store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the AnswersetName parameter with the Search-Record-Id (10 bytes). The maximum AnswersetName length is 22 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the recid of the record to start searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the recid of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Prototype:

Imports ssa

```
Public Function search_fields_count ( _  
    ByRef searchname As String _' String in  
) As Integer
```

Parameters:

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Prototype:

Imports ssa

```
Public Function search_fields_get ( _  
    ByRef searchname As String _' String in  
) As String ()
```

Parameters:

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the *SQL Filters* section of the DESIGNER MANUAL for details about SQL filters

Prototype:

Imports ssa

```
Public Sub search_filter ( _  
    ByRef search As String, _' String in  
    ByRef filter As String _' String in  
)
```

Parameters:

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

search_finish

Description:

Release resources associated with `ids_search_start`.

Prototype:

Imports ssa

```
Public Sub search_finish ( _  
    ByRef search As String _ ' String in  
)
```

Parameters:

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Prototype:

Imports ssa

```
Public Function search_get ( _  
    ByRef searchname As String _ ' String in  
) As search_get_struct
```

Parameters:

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, frep[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and SSA-NAME3 v1 is used.

Prototype:

Imports ssa

```
Public Function search_get_complete ( _  
    ByRef search As String _' String in  
) As search_get_complete_struct
```

Parameters:

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4 \times (1 + 3 \times 100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Prototype:

Imports ssa

```
Public Function search_get_detail ( _  
    ByRef search As String _' String in  
) As search_get_detail_struct
```

Parameters:

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Prototype:

Imports ssa

```
Public Function search_IDT_get ( _  
    ByRef searchname As String _' String in  
) As String
```

Parameters:

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Prototype:

Imports ssa

```
Public Function search_layout ( _  
    ByRef search As String, _' String in  
    ByRef viewType As String, _' String in  
    ByRef func As String _' String in  
) As search_layout_struct
```

Parameters:

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the FORMATS_SIZE constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

search_profile_count

Description:

Count the number of profile entries available

Prototype:

Imports ssa

Public Function search_profile_count () **As Integer**

Parameters:

count Returns the number of elements

Return Code:

negative for error, 0 for success

search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Prototype:

Imports ssa

Public Function search_profile_get () **As** search_profile_get_struct

Parameters:

ids the numeric identifiers for each field, valid value are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Prototype:

Imports ssa

```
Public Function search_start ( _  
    ByRef search As String, _' String in  
    ByRef search_width As String, _' String in  
    ByRef match_tolerance As String, _' String in  
    ByRef parameters()() As Byte, _' BlockArray in  
    ByVal searchrec() As Byte, _' Block io  
    ByRef searchrec_size As Integer, _  
    ByRef AnswersetName As String, _' String in  
    ByRef records()() As Byte _' BlockArray in  
) As Integer
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an `AnswerSet`. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an `AnswerSet`, set `AnswersetName` to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding `SEARCH_LIMIT` and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

search_start_via_parameters

Description:

Prototype:

Imports ssa

```
Public Function search_start_via_parameters ( _  
    ByRef search As String, _' String in  
    ByRef search_width As String, _' String in  
    ByRef match_tolerance As String, _' String in  
    ByRef parameters()() As Byte _' BlockArray in  
) As search_start_via_parameters_struct
```

Parameters:

search is the name of the search to be performed.

search_width specifies either *Narrow*, *Typical* or *Exhaustive* to nominate how many candidates should be selected. If left blank, the *Search_Level* in the Controls will be used.

match_tolerance specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the *Match_Level* in the Controls will be used.

parameters contains the field values used to construct a search record (*searchrec*). The order of the field values must correspond to the order of fields returned by *ids_search_fields_get*. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the *Unique_Keys* option to remove duplicates, the resulting set may contain less than *recs* records. In this case, you must use the response code from *ids_search_get* to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

search_start_via_record

Description:

Prototype:

Imports ssa

```
Public Function search_start_via_record ( _  
    ByRef search As String, _' String in  
    ByRef search_width As String, _' String in  
    ByRef match_tolerance As String, _' String in  
    ByRef searchrec() As Byte _' Block in  
) As search_start_via_record_struct
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

Imports ssa

```
Public Function search_tolerances_count ( _  
    ByRef searchname As String _' String in  
) As Integer
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Prototype:

Imports ssa

```
Public Function search_tolerances_get ( _  
    ByRef searchname As String _' String in  
) As String ()
```

Parameters:

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: **view_field_count** is needed to dynamically allocate the arrays used for calls to **ids_search_layout** and **view_length** is used to dynamically allocate memory for input/output records.

Prototype:

Imports ssa

```
Public Function search_view_get ( _  
    ByRef search As String, _' String in  
    ByRef viewType As String _' String in  
) As search_view_get_struct
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

search_view_set

Description:

Sets a view as the active input or output view

Prototype:

```
Imports ssa

Public Sub search_view_set ( _
    ByRef search As String, _' String in
    ByRef viewType As String, _' String in
    ByRef viewName As String _' String in
)
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

```
Imports ssa

Public Function search_widths_count ( _
    ByRef searchname As String _' String in
) As Integer
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Prototype:

Imports ssa

```
Public Function search_widths_get ( _  
    ByRef searchname As String _' String in  
) As String ()
```

Parameters:

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

server_version_get

Description:

Get the version information associated with the server.

Prototype:

Imports ssa

```
Public Function server_version_get () As String
```

Parameters:

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Prototype:

```
Imports ssa  
  
Public Sub session_close ()
```

Parameters:

none

Return Code:

negative for error, 0 for success

session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Prototype:

```
Imports ssa  
  
Public Sub session_open ( _  
    ByVal session As Integer _ ' Long io  
)
```

Parameters:

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the *Globalization* section of the OPERATIONS manual for further details and restrictions.

Prototype:

Imports ssa

```
Public Sub set_encoding ( _  
    ByRef encoding As Integer _' Long in  
)
```

Parameters:

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

set_timeout

Description:

Informs the Search Server of the timeout value in seconds The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached)

Prototype:

Imports ssa

```
Public Sub set_timeout ( _  
    ByRef timeout As Integer _' Long in  
)
```

Parameters:

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, **VPD** section for details about VPD.

Prototype:

Imports ssa

```
Public Sub set_vpd_user ( _  
    ByRef vpd_user As String, _' String in  
    ByRef vpd_ctx As String _' String in  
)
```

Parameters:

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

system_close

Description:

closes the system and frees any remaining resources.

Prototype:

Imports ssa

```
Public Sub system_close ()
```

Parameters:

none

Return Code:

negative for error, 0 for success

system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Prototype:

Imports ssa

Public Function system_idtname_count () **As Integer**

Parameters:

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Prototype:

Imports ssa

Public Function system_idtname_get () **As String** ()

Parameters:

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

system_notify

Description:

Notifies search server on a system.

Prototype:

Imports ssa

```
Public Sub system_notify ( _  
    ByRef rulebase As String, _' String in  
    ByRef sysname As String, _' String in  
    ByRef message As String _' String in  
)
```

Parameters:

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

system_open

Description:

opens a system.

Prototype:

Imports ssa

```
Public Sub system_open ( _  
    ByRef rulebase As String, _' String in  
    ByRef system As String, _' String in  
    ByRef verbosity As String, _' String in  
    ByRef Options As String _' String in  
)
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

system_search_finish

Description:

Finishes the search and closes the system.

Prototype:

Imports ssa

Public Sub system_search_finish ()

Parameters:

none

Return Code:

negative for error, 0 for success

system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Prototype:

Imports ssa

```
Public Function system_search_start ( _  
    ByRef rulebase As String, _' String in  
    ByRef system As String, _' String in  
    ByRef verbosity As String, _' String in  
    ByRef options As String, _' String in  
    ByRef search As String, _' String in  
    ByRef parameters() As Byte, _' BlockArray in  
    ByRef AnswersetName As String _' String in  
) As system_search_start_struct
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1, KEYWORD2=VALUE2,

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

datalen will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the SHOWALLSEARCHES keyword to the option string of ids_system_open.

Prototype:

Imports ssa

Public Function system_searches_count () **As Integer**

Parameters:

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the SHOWALLSEARCHES keyword to the option string of ids_system_open.

Prototype:

Imports ssa

Public Function system_searches_get () **As String** ()

Parameters:

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

systems_count

Description:

the number of systems in the rulebase.

Prototype:

Imports ssa

Public Function systems_count (_
 ByRef rulebase **As String** _' *String in*
) **As Integer**

Parameters:

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

systems_get

Description:

Get the names of all the systems defined in the rulebase.

Prototype:

Imports ssa

```
Public Function systems_get ( _  
    ByRef rulebase As String _' String in  
) As String ()
```

Parameters:

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Using IIR with XML

XML is not another language binding but a different way of talking to Identity Resolution.

WSDL

A Web Services Description Language (WSDL) file for the search API is created in the server work directory when the server starts or is refreshed.

The WSDL can also be accessed through the server at:

```
http://<xmhost>:<xmport>/?search.wsdl
```

Creating a proxy with .NET

A proxy can be created from the WSDL generated by the XML Search Server using `wsdl.exe`, which is part of the Microsoft .NET SDK. Given the WSDL, one can create a proxy with:

```
wsdl /out:search.cs search.wsdl
```

This creates a C# `public class` called `search` which can be used to access the XML search API.

Creating a proxy with Apache Axis2

A proxy can be created from the WSDL generated by the XML Search Server using Apache Axis2. Given the WSDL, one can create a proxy on Windows with:

```
%AXIS2_HOME%\bin\wsdl2java -uri search.wsdl -d adb -s -p search
```

or on Unix with:

```
sh $AXIS2_HOME/bin/wsdl2java.sh -uri search.wsdl -d adb -s -p search
```

This creates a Java `public class` called `search` which can be used to access the XML search API.

XML and HTTP

XML messages are sent to the Search Server using Hypertext Transfer Protocol (HTTP), the protocol used by Web servers. See the *Servers / Starting* section in the OPERATIONS manual for more information on enabling the HTTP/XML protocol.

The general format of an HTTP request looks like this:

request-line

headers (0 or more)

<blank line>

body

A simple HTTP request header looks like this:

```
POST / HTTP/1.1
TE: deflate,gzip;q=0.3
Connection: TE, close
Host: ssa:1665
User-Agent: libwww-perl/5.814
Content-Length: 1044
```

A simple HTTP response header looks like this:

```
HTTP/1.1 200 OK
Date: Thu, 04 Mar 2010 03:11:49 GMT
User-Agent: Informatica-IR/9.0.01
Content-Type: text/xml; charset=UTF-8
Cache-Control: no-cache
SOAPAction: ""
Content-Length: 1149
```

While the last figure represents the length of the body in bytes.

XML and SOAP

The body must be formatted as a Simple Object Access Protocol (SOAP) request. The function call block identifies the function and its parameters. These are provided in the form of XML tags. Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <ps_set_population xmlns="http://www.identitysystems.com/xmlschema/iss-version-9/se"
      <session>452984832</session>
      <system>default</system>
      <population>std</population>
    </ps_set_population>
  </soap:Body>
</soap:Envelope>
```

A response will look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/03/addressing"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-s"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-ut"
  <soap:Header>
    <wsa:MessageID>urn:uuid:3f1d7c19-b7f2-4432-a4ef-469a11926c91</wsa:MessageID>
    <wsa:Action>ps_set_population</wsa:Action>
    <wsa:To>http://www.w3.org/2005/03/addressing/role/anonymous</wsa:To>
    <wsa:From>
      <wsa:Address>http://ssa:1665</wsa:Address>
    </wsa:From>
```

```

    <wsse:Security>
      <wsu:Timestamp wsu:Id="Timestamp-d3ca2448-9c81-4427-8701-847e632fdc7f">
        <wsu:Created>2010-03-04T04:11:49Z</wsu:Created>
        <wsu:Expires>2010-03-04T04:16:49Z</wsu:Expires>
      </wsu:Timestamp>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <ps_set_population_response xmlns="http://www.identitysystems.com/xmlschema/iss-ver
      <response>0</response>
      <session>452984832</session>
    </ps_set_population_response>
  </soap:Body>
</soap:Envelope>

```

An error will produce a SOAP fault. For example:

```

<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:wsa="http://www.w3.org/2005/03/addressing"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-s
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsswssecurity-ut
<soap:Header>
  <wsa:MessageID>urn:uuid:e231f126-b369-4113-8bfa-4c2932126630</wsa:MessageID>
  <wsa:Action>http://www.w3.org/2005/03/addressing/fault</wsa:Action>
  <wsa:RelatesTo wsa:RelationshipType="http://www.w3.org/2005/08/addressing/reply"
    >urn:uuid:45e4ac46-c7ce-4c0a-b2e0-8410665090c4</wsa:RelatesTo>
  <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
  <wsa:From>
    <wsa:Address>http://ssa:1665</wsa:Address>
  </wsa:From>
  <wsse:Security>
    <wsu:Timestamp wsu:Id="Timestamp-4d52859b-5f59-4bab-b88b-c5925860a394">
      <wsu:Created>2010-03-04T06:39:50Z</wsu:Created>
      <wsu:Expires>2010-03-04T06:44:50Z</wsu:Expires>
    </wsu:Timestamp>
  </wsse:Security>
</soap:Header>
<soap:Body>
  <soap:Fault>
    <soap:faultcode>Client</soap:faultcode>
    <soap:faultstring>user_open: RuleBase name is blank</soap:faultstring>
    <soap:detail>
      <ssa:error xmlns:ssa="http://www.identitysystems.com/xmlschema/iss-version-9">
        <ssa:response>-11010109</ssa:response>
        <ssa:function>system_search_start</ssa:function>
        <ssa:session>721420288</ssa:session>
        <ssa:details>[304163950 17] ssasrsv &gt; It is now 20100304163950
[304163950 17] ssaxmsl.c      871 rc  9  40110101*100
[304163950 17] ssasesi.c    4244 rc  1  10401101*100
[304163950 17] ssasesi.c    1363 rc  1  104011*100
[304163950 17] open system 'testx228' in rulebase failed. Check the Rulebase status
[304163950 17] ssarbcx.c     735 rc  1  10401*10
[304163950 17] ssaru.c       617 rc  1  104*100
[304163950 17] ssaru.c       594 rc  4  1*100
[304163950 17] ssaru.c       514 rc  1
[304163950 17] user_open: RuleBase name is blank
</ssa:details>
      </ssa:error>
    </soap:detail>
  </soap:Fault>
</soap:Body>
</soap:Envelope>

```

```

        </soap:detail>
    </soap:Fault>
</soap:Body>
</soap:Envelope>

```

XML and Unicode

XML messages must be sent in Unicode, either in UTF-8 or UTF-16. No other character sets are supported.

Any message sent in UTF-8 will be replied to in UTF-8, while one sent in UTF-16 is replied to in UTF-16. All strings must be valid Unicode strings. This includes search names and the like. However blocks may contain any character set. Refer to the OPERATIONS manual, *Globalization* chapter for more details about issues relating to multiple languages, character sets and UNICODE.

Parameter types

Note: values in **BOLD** represent information that must be provided to the function.

ids_addr_get_cass_field

Description:

Use this function to retrieve a validated CASS field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_cass_field
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <field_idx>301</field_idx>
      <field_value_size>50</field_value_size>
    </ids_addr_get_cass_field>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_cass_field_response

```

```

xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
  <response>0</response>
  <session>3436273</session>
  <field_value>value</field_value>
</ids_addr_get_cass_field_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a cass field

field_idx Specifies a cass field within the nth suggestion

field_value The cass field value

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_cnt

Description:

This function returns the maximum number of CASS address fields created as a result of a parse or validate call. Use this value to dynamically allocate the `field_length` array for the `ids_addr_parse` API.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_cass_field_cnt
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_addr_get_cass_field_cnt>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_cass_field_cnt_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <count>302</count>
    </ids_addr_get_cass_field_cnt_response>
  </soap:Body>
</soap:Envelope>

```

```

    </ids_addr_get_cass_field_cnt_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

count Returns the max number of cass address fields

Return Code:

negative for error, 0 for success

ids_addr_get_cass_field_info

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_cass_field_info
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <field_length_num>1</field_length_num>
    </ids_addr_get_cass_field_info>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_cass_field_info_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <field_lengthArray>
        <field_length>301</field_length>
      </field_lengthArray>
    </ids_addr_get_cass_field_info_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each cass address field

Return Code:

negative for error, 0 for success

ids_addr_get_del_lines_ext

Description:

Use this function to retrieve delivery address line information

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_del_lines_ext
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <del_case>301</del_case>
      <del_line1_size>50</del_line1_size>
      <del_line2_size>50</del_line2_size>
      <del_line3_size>50</del_line3_size>
      <del_line4_size>50</del_line4_size>
      <del_line5_size>50</del_line5_size>
      <del_line6_size>50</del_line6_size>
    </ids_addr_get_del_lines_ext>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_del_lines_ext_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <del_line1>value</del_line1>
      <del_line2>value</del_line2>
      <del_line3>value</del_line3>
      <del_line4>value</del_line4>
      <del_line5>value</del_line5>
      <del_line6>value</del_line6>
    </ids_addr_get_del_lines_ext_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

suggest_idx Specifies the suggestion from which to get delivery address lines

del_case Specifies delivery address line case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

del_line1 delivery address line 1 output string

del_line2 delivery address line 2 output string

del_line3 delivery address line 3 output string

del_line4 delivery address line 4 output string

del_line5 delivery address line 5 output string

del_line6 delivery address line 6 output string

Return Code:

negative for error, 0 for success

ids_addr_get_field

Description:

Use this function to retrieve a validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`). `val_status` and `val_mods` return a code that describes how the field matched to validation data and whether or not it was modified by validation. Refer to the *Address Validation* section of this manual for a list of codes.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <field_idx>301</field_idx>
      <field_value_size>50</field_value_size>
    </ids_addr_get_field>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
```

```

xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <field_value>value</field_value>
      <field_val_status>302</field_val_status>
      <field_val_mods>302</field_val_mods>
    </ids_addr_get_field_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field

field_idx Specifies a field within the nth suggestion

field_value The field value

field_val_status Specifies how this field matched the validation data

field_val_mods Specifies how this field was modified by validation data

Return Code:

negative for error, 0 for success

ids_addr_get_field_count

Description:

This function returns the maximum number of address fields created as a result of a parse or validate call. Use this value to dynamically allocate the field_length array for the ids_addr_parse API.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_addr_get_field_count>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <count>302</count>
    </ids_addr_get_field_count_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

count Returns the max number of address fields

Return Code:

negative for error, 0 for success

ids_addr_get_field_ext

Description:

Use this function to retrieve all getter fields

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_ext
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <field_operation>301</field_operation>
      <field_name>value</field_name>
      <field_item_line>301</field_item_line>
      <field_type>value</field_type>
      <field_value_size>50</field_value_size>
    </ids_addr_get_field_ext>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

```

```

<soap:Body>
  <ids_addr_get_field_ext_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
    <response>0</response>
    <session>3436273</session>
    <field_value>value</field_value>
  </ids_addr_get_field_ext_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

suggest_idx Specifies the suggestion from which to get fields

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines Option 2 for AddressComplete Option 3 for EnrichmentData Option 4 for ResultDataParameter Option 5 for EnrichmentDataStatus

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value Cleansed field output

Return Code:

negative for error, 0 for success

ids_addr_get_field_idx

Description:

Use this function to retrieve a parsed or validated field. The `suggestion_idx` specifies the suggestion from which to select the field value. (0 for parsed data, 1..n for validated data, where n is the `n_suggest` parameter returned by `ids_addr_validate`).

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_idx
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <field_idx>301</field_idx>
      <field_value_size>50</field_value_size>
    </ids_addr_get_field_idx>
  </soap:Body>
</soap:Envelope>

```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_idx_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <field_value>value</field_value>
    </ids_addr_get_field_idx_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

suggest_idx Specifies the nth suggestion from which to get a field. On successful parse, use 0 for ASM/AD version 4, 1 for ASM/AD version 5

field_idx Specifies a field within the nth suggestion

field_value The field value

Return Code:

negative for error, 0 for success

ids_addr_get_field_info_ext

Description:

Use this function to retrieve a list of individual field lengths after validating an address. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_info_ext
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <suggest_idx>301</suggest_idx>
      <field_length_num>1</field_length_num>
      <addr_label_encoded_size>50</addr_label_encoded_size>
      <addr_label_charset_size>256</addr_label_charset_size>
    </ids_addr_get_field_info_ext>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_info_ext_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <field_lengthArray>
        <field_length>301</field_length>
      </field_lengthArray>
      <addr_label_encoded>value</addr_label_encoded>
      <addr_label_charset>value</addr_label_charset>
      <score>302</score>
    </ids_addr_get_field_info_ext_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

suggest_idx Specifies the suggestion from which to retrieve information

field_length An array containing the length of each address field

addr_label_encoded The returned label

addr_label_charset The character set used in the address label

score The returned label's score

Return Code:

negative for error, 0 for success

ids_addr_get_field_len

Description:

This function returns the maximum length of an individual address field. It may be used to dynamically allocate the field parameter used for the `ids_addr_get_field_idx` API.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_len
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_addr_get_field_len>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_field_len_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <max_len>302</max_len>
    </ids_addr_get_field_len_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

max_len Returns the max address field length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_line_len

Description:

This function returns the maximum length of an input address line. It may be used to dynamically allocate the input lines used for the `ids_addr_set_lines` API.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_line_len
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_addr_get_line_len>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_line_len_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
```

```

    <response>0</response>
    <session>3436273</session>
    <max_len>302</max_len>
  </ids_addr_get_line_len_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

max_len Returns the max line length in bytes

Return Code:

negative for error, 0 for success

ids_addr_get_option

Description:

Use this function to obtain values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_option
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <param>value</param>
      <value_size>256</value_size>
    </ids_addr_get_option>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_get_option_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <value>value</value>
    </ids_addr_get_option_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

param This field specifies the name of the option to get.

value Returns the value of the option.

Return Code:

negative for error, 0 for success

ids_addr_info

Description:

Use this function to request additional information about an input address. This call must always be preceded with a call to `ids_addr_std`.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_info
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <controls>value</controls>
      <value_size>256</value_size>
    </ids_addr_info>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_info_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <value>value</value>
    </ids_addr_info_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

controls this field contains the request information. It must be specified in the form ITEM=[value].

value this field contains the requested information.

Return Code:

negative for error, 0 for success

ids_addr_init

Description:

This function initializes the Address Standardization sub-system. It must be the first call to `ids_addr_*` family of functions. The `max_memory` parameter specifies the maximum amount of memory (MB) to be used by the Address Standardization engine (within the Search Server process).

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_init
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <max_memory>301</max_memory>
    </ids_addr_init>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_init_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_init_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

max_memory This field specifies the maximum amount of memory (MB) to be used by the Address Standardization engine.

Return Code:

negative for error, 0 for success

ids_addr_parse

Description:

Use this function to parse an address. The individual field lengths after parsing an address are returned in the `field_length` array. Fields with a length of zero have no value associated with them and can be omitted from the list of fields retrieved with `ids_addr_get_field_idx`

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_parse
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <field_length_num>1</field_length_num>
    </ids_addr_parse>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_parse_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <field_lengthArray>
        <field_length>301</field_length>
      </field_lengthArray>
    </ids_addr_parse_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

field_length An array containing the length of each parsed field

Return Code:

negative for error, 0 for success

ids_addr_preload_country

Description:

Use this function to preload country database

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_preload_country
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <preload_type>value</preload_type>
      <preload_country>value</preload_country>
      <val_mode>value</val_mode>
    </ids_addr_preload_country>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_preload_country_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_preload_country_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

preload_type Type of preload to perform

preload_country Country database to be preloaded

val_mode Validation mode to be used

Return Code:

negative for error, 0 for success

ids_addr_set_attrb

Description:

Use this function to specify the character set of the data (for both input and output). The default_country parameter specifies that default country to use when parsing cannot identify a country from the address. This API must be called prior to parsing or validating an address. The values stay in effect for the life of the session, or until they are changed.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_attrib
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <char_set>value</char_set>
      <default_country>value</default_country>
    </ids_addr_set_attrib>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_attrib_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_attrib_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

char_set The name of the character set used to encode the input and output.

default_country The default country used for validation when parsing cannot detect a country name.

Return Code:

negative for error, 0 for success

ids_addr_set_del_lines

Description:

Use this function to set delivery address line information

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```

<soap:Body>
  <ids_addr_set_del_lines
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
    <session>3436273</session>
    <del_line1>value</del_line1>
    <del_line1_size>50</del_line1_size>
    <del_line2>value</del_line2>
    <del_line2_size>50</del_line2_size>
    <del_line3>value</del_line3>
    <del_line3_size>50</del_line3_size>
    <del_line4>value</del_line4>
    <del_line4_size>50</del_line4_size>
    <del_line5>value</del_line5>
    <del_line5_size>50</del_line5_size>
    <del_line6>value</del_line6>
    <del_line6_size>50</del_line6_size>
  </ids_addr_set_del_lines>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_del_lines_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_del_lines_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

del_line1 delivery address line 1 input string
del_line2 delivery address line 2 input string
del_line3 delivery address line 3 input string
del_line4 delivery address line 4 input string
del_line5 delivery address line 5 input string
del_line6 delivery address line 6 input string

Return Code:

negative for error, 0 for success

ids_addr_set_field_case

Description:

Use this function to set individual input fields case option

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_case
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <field_idx>301</field_idx>
      <field_case>301</field_case>
    </ids_addr_set_field_case>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_case_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_field_case_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

field_idx Specifies the nth field to set

field_case Specifies output field case option value. The allowed values are 0 = Unchanged, 1 = Upper case, 2 = Lower case and 3 = Mixed case.

Return Code:

negative for error, 0 for success

ids_addr_set_field_ext

Description:

Use this function to set fields

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
```

```

<soap:Body>
  <ids_addr_set_field_ext
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
    <session>3436273</session>
    <field_operation>301</field_operation>
    <field_name>value</field_name>
    <field_item_line>301</field_item_line>
    <field_type>value</field_type>
    <field_value>value</field_value>
    <field_value_size>50</field_value_size>
  </ids_addr_set_field_ext>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_ext_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_field_ext_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

field_operation Field operation Option 0 for AddressElements Option 1 for AddressLines

field_name Refer AD Result.dtd for field names

field_item_line Represent field line number or field item number

field_type Refer AD Result.dtd for field attribute Type

field_value input field value

Return Code:

negative for error, 0 for success

ids_addr_set_field_idx

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_idx
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <field_idx>301</field_idx>
      <field_value>value</field_value>
      <field_value_size>50</field_value_size>
    </ids_addr_set_field_idx>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_idx_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_field_idx_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

field_idx Specifies the nth field to set

field_value Specifies a value for the nth field

Return Code:

negative for error, 0 for success

ids_addr_set_field_name

Description:

Use this function to set individual input fields by name

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_name

```

```

xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
  <session>3436273</session>
  <field_name>value</field_name>
  <field_value>value</field_value>
  <field_value_size>50</field_value_size>
</ids_addr_set_field_name>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_field_name_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_field_name_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

field_name Specifies the name of the field to set

field_value Specifies a value for the field

Return Code:

negative for error, 0 for success

ids_addr_set_lines

Description:

Use this function to specify the value of an input field. This API is used to specify an input address that has already been pre-parsed into separate fields.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_lines
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <line_1>value</line_1>
      <line_1_size>50</line_1_size>
      <line_2>value</line_2>

```

```

    <line_2_size>50</line_2_size>
    <line_3>value</line_3>
    <line_3_size>50</line_3_size>
    <line_4>value</line_4>
    <line_4_size>50</line_4_size>
    <line_5>value</line_5>
    <line_5_size>50</line_5_size>
    <line_6>value</line_6>
    <line_6_size>50</line_6_size>
    <line_7>value</line_7>
    <line_7_size>50</line_7_size>
    <line_8>value</line_8>
    <line_8_size>50</line_8_size>
    <line_9>value</line_9>
    <line_9_size>50</line_9_size>
    <line_10>value</line_10>
    <line_10_size>50</line_10_size>
  </ids_addr_set_lines>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_lines_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_lines_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

line_1 The first line of the address

line_2 The second line of the address

line_3 The third line of the address

line_4 The fourth line of the address

line_5 The fifth line of the address

line_6 The sixth line of the address

line_7 The seventh line of the address

line_8 The eighth line of the address

line_9 The ninth line of the address

line_10 The tenth line of the address

Return Code:

negative for error, 0 for success

ids_addr_set_option

Description:

Use this function to set values of options that control Address Standardization behavior. A list of options appears in the *Address Standardization* section of this manual.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_option
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <param>value</param>
      <value>value</value>
    </ids_addr_set_option>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_set_option_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_addr_set_option_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

param This field specifies the name of the option to set.

value This field specifies a value for the option.

Return Code:

negative for error, 0 for success

ids_addr_std

Description:

Use this function to request IDS to standardize an address by validating it against USPS validation tables and formatting it to comply with U.S. Postal Addressing Standards. This API requires the separately licensable *IDS Address Standardization Module* to be installed.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_std
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <firm_name>value</firm_name>
      <firm_name_size>256</firm_name_size>
      <urbanization>value</urbanization>
      <urbanization_size>256</urbanization_size>
      <address_one>value</address_one>
      <address_one_size>256</address_one_size>
      <address_two>value</address_two>
      <address_two_size>256</address_two_size>
      <last_line>value</last_line>
      <last_line_size>256</last_line_size>
    </ids_addr_std>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_std_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <firm_name>value</firm_name>
      <urbanization>value</urbanization>
      <address_one>value</address_one>
      <address_two>value</address_two>
      <last_line>value</last_line>
    </ids_addr_std_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

firm_name It contains the name of the firm (may be blank).

urbanization this field can contain name of an urban development within a geographic area. It is only used with Puerto Rican addresses.

address_one this field contains the Delivery Address Line. It normally consists of a street number, pre-directional, street name, street suffix, post-directional and possibly some secondary address components such as apartment number.

address_two this field contains additional Delivery Address Line components. It is normally only used when address_one is very long.

last_line this field contains the Last Line information: the city name, state abbreviation and zip code (and possibly the Zip + 4 code).

Return Code:

- 0 indicates an exact match to a valid address
- 1 indicates a no match (invalid address)
- 2 indicates a multi match (non-unique address), and
- < 0 indicates an error

ids_addr_validate

Description:

Use this function to validate an address

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_validate
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_addr_validate>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_addr_validate_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <status>302</status>
      <n_suggest>302</n_suggest>
    </ids_addr_validate_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

status The status returned by the validation process

n_suggest The number of suggestions generated by validation

Return Code:

negative for error, 0 for success

ids_connect

Description:

Initiates a socket.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_connect
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <host>value</host>
      <port>301</port>
    </ids_connect>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_connect_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <sockh>302</sockh>
    </ids_connect_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

host is the host to connect to.

port is the port to connect to.

sockh is a socket handle.

Return Code:

negative for error, 0 for success

ids_disconnect

Description:

Releases resources allocated to a socket.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_disconnect
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_disconnect>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_disconnect_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_disconnect_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_error_get

Description:

Get the error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, *Error Log* section for help in interpreting the Error Log.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
```

```

xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_error_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <msg_size>256</msg_size>
    </ids_error_get>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_error_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <msg>value</msg>
    </ids_error_get_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

msg is the error message returned

Return Code:

0 for success, -ve for error and 1 for no more errors to retrieve.

ids_errors_get_all

Description:

Get the Server side error messages from the last API function that failed. This function should be called repeatedly until it returns 1, meaning all messages have been retrieved.

Note: if a communication (socket) error occurred, this function will also fail. Refer to the OPERATIONS MANUAL, **Error Log** section for information on interpreting the Error Log.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_errors_get_all

```

```

xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
  <session>3436273</session>
  <msg_size>256</msg_size>
</ids_errors_get_all>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_errors_get_all_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <msg>value</msg>
    </ids_errors_get_all_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

msg is an error message.

Return Code:

negative for error, 0 for success

ids_identify

Description:

Identify a session to the console

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_identify
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <identification>value</identification>
    </ids_identify>
  </soap:Body>
</soap:Envelope>

```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_identify_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_identify_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

identification is user supplied identification for display on the console

Return Code:

negative for error, 0 for success

ids_is_little_endian

Description:

Checks if the search server is running on a little endian platform

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_is_little_endian
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_is_little_endian>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_is_little_endian_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <endian_state>302</endian_state>
    </ids_is_little_endian_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

endian_state Returns 1 if the search server is running on a little endian platform. Returns 0 for others

Return Code:

negative for error, 0 for success

ids_match_explain

Description:

Explains the match result given search and file records As match_explain_count does not give the exact number of output rows for this call, but instead provides a maximal estimate, some of the info blocks returned will be filled with NULL bytes. Test a block for validity by checking the first byte is not NULL. Info blocks returned are not all the same length either. They are NULL filled on the right.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_match_explain
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <match_tolerance>value</match_tolerance>
      <searchrec>value</searchrec>
      <searchrec_size>50</searchrec_size>
      <filerec>value</filerec>
      <filerec_size>50</filerec_size>
      <info_array_num>1</info_array_num>
      <info_array_size>50</info_array_size>
    </ids_match_explain>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_match_explain_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <info_arrayArray>
        <info_array>value</info_array>
      </info_arrayArray>
    </ids_match_explain_response>
  </soap:Body>
</soap:Envelope>
```

```

    </ids_match_explain_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the Search which was performed.

match_tolerance specifies either *Conservative*, *Typical* or *Loose* to nominate how aggressive the matching scheme should be in rejecting candidates.

searchrec is the original record that we searched on

filerec is the record that was returned by the search

info_array An array describing the match results. See the *Match Explain API* section for details

Return Code:

negative for error, 0 for success

ids_match_explain_count

Description:

Estimate number of info blocks required for a subsequent `match_explain` call

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_match_explain_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
    </ids_match_explain_count>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_match_explain_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <count>302</count>
    </ids_match_explain_count_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the Search which was performed.

count Returns the maximum number of info blocks required to explain the search results

Return Code:

negative for error, 0 for success

ids_real_time_async_get

Description:

Used to retrieve the result count associated with a call to `ids_real_time_async_start`. `cluster_action_count` specifies the number of results that are available for collection using `ids_real_time_sync_get`.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_async_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <reference>value</reference>
      <block>301</block>
    </ids_real_time_async_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_async_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <cluster_action_count>302</cluster_action_count>
    </ids_real_time_async_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

reference A reference number identifying the request (returned by `ids_real_time_sync_start`)

block 1 = wait for a response 0 = return immediately if no results are available yet

cluster_action_count The number of clusters generated These can be returned with call to `real_time_sync_get` 1 = wait for a response

Return Code:

negative for error 0 for success positive for no results are available yet

ids_real_time_async_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in an asynchronous fashion and will return when the transaction has been placed on the server's work queue rather than when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by a call to `ids_real_time_async_get`, then by one or more calls to `ids_real_time_sync_get` to retrieve the results.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_async_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
      <system>value</system>
      <IDT>value</IDT>
      <sequence_number>value</sequence_number>
      <operation>value</operation>
      <cluster_record>value</cluster_record>
      <cluster_record_size>50</cluster_record_size>
      <source>301</source>
      <multi_search>value</multi_search>
      <input_id>301</input_id>
      <reference_size>256</reference_size>
    </ids_real_time_async_start>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_async_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
```

```

    <session>3436273</session>
    <reference>value</reference>
  </ids_real_time_async_start_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

source Identifies the source of clustering: 0 = Real Time Synchronizer 1 = Flat file 2 = NSA table

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT/IDX only transaction.

input_id Reserved. A value of 0 must be passed for this parameter.

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success positive for warnings:

1-9 Reserved for future use

10 warnings: Duplicate PK detected on add to IDT with `WARN_DUPLICATE_PK` sync option.

11 warnings: Duplicate transaction was skipped.

12 warnings: Transaction was rejected because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

ids_real_time_flul_add

Description:

This API used to add force link and unlink rule. This call must be followed by `ids_real_time_flul_init`.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_add
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rule_type>value</rule_type>
      <subject_rec_pk>value</subject_rec_pk>
      <subject_rec_pk_size>50</subject_rec_pk_size>
      <relationship>value</relationship>
      <related_rec_pk>value</related_rec_pk>
      <related_rec_pk_size>50</related_rec_pk_size>
    </ids_real_time_flul_add>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_add_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_real_time_flul_add_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

rule_type This field is for specifying the type of the rule. A value of 'A' represents that the rule needs to be added to the system and 'D' represents that a rule needs to be removed from the system

subject_rec_pk This field is for specifying the PK of the subject record

relationship This field is for specifying the relationship between the subject record and related record. A value of 'L' represents a Link rule between the subject record and the related record and a value of 'U' represents an Unlink rule between the subject record and the related record.

related_rec_pk This field is for specifying the PK of the record that is either linked or unlinked to the subject record.

Return Code:

negative for error, 0 for success 3 when the Link rule is not allowed. Record unlinked with members of subject cluster.

ids_real_time_flul_close

Description:

This API used to close and release force link and unlink module.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_close
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_real_time_flul_close>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_close_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_real_time_flul_close_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_real_time_flul_delete

Description:

This API used to delete force link and unlink rule from MR table. This call must be followed by ids_real_time_flul_init.

Input message:

```
<?xml version="1.0"?>
```

```

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_delete
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rule_type_option>301</rule_type_option>
      <record_pk>value</record_pk>
      <record_pk_size>50</record_pk_size>
    </ids_real_time_flul_delete>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_delete_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_real_time_flul_delete_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

rule_type_option 0 Delete only disabled rules for input pk.

1 Delete only active rules for input pk.

2 Delete all rules for input pk.

record_pk This field is for specifying the PK of the record to be deleted

Return Code:

negative for error, 0 for success

ids_real_time_flul_find_rule

Description:

This API used to find link and unlink information for input IDT record. should be called after ids_real_time_flul_init API.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

```

```

<soap:Body>
  <ids_real_time_flul_find_rule
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
    <session>3436273</session>
    <idt_rec>value</idt_rec>
    <idt_rec_size>50</idt_rec_size>
    <option>301</option>
  </ids_real_time_flul_find_rule>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_find_rule_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_real_time_flul_find_rule_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

idt_rec This field is for specifying the PK of the record to be searched

option 0 Link Rule.

1 Unlink Rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_get_rule

Description:

This API used fetch link and unlink information for input IDT record. Should be called after `ids_real_time_flul_find_rule` API.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_get_rule
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>

```

```

        <idt_rec_size>50</idt_rec_size>
    </ids_real_time_flul_get_rule>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_get_rule_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <idt_rec>value</idt_rec>
    </ids_real_time_flul_get_rule_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

idt_rec is the matched File record for input link or unlink rule.

Return Code:

negative for error, 0 for success

ids_real_time_flul_init

Description:

This API used to initialize force link and unlink module. Memory allocated a part of `ids_real_time_flul_init` is released using `ids_real_time_flul_close`.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_init
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <idt_name>value</idt_name>
      <multi_search>value</multi_search>
    </ids_real_time_flul_init>
  </soap:Body>
</soap:Envelope>

```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_flul_init_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_real_time_flul_init_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

idt_name is the name of the IDT associated with the force link and unlink rule.

multi_search The name of the multi-search which uses Persistent-ID. This parameter should be set to an empty string for an IDT only MR rule creation.

Return Code:

negative for error, 0 for success

ids_real_time_sync_get

Description:

Use to retrieve the results and free the resources associated with a call to `ids_real_time_sync_start` or `ids_real_time_async_start`. Should be called until it returns a non zero response.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_sync_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <reference>value</reference>
      <cluster_action_type_size>256</cluster_action_type_size>
      <cluster_action_id_size>256</cluster_action_id_size>
      <cluster_action_new_size>256</cluster_action_new_size>
    </ids_real_time_sync_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
```

```

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_sync_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <cluster_action_type>value</cluster_action_type>
      <cluster_action_id>value</cluster_action_id>
      <cluster_action_number>302</cluster_action_number>
      <cluster_action_new>value</cluster_action_new>
    </ids_real_time_sync_get_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

cluster_action_type Identifies the action, ie add or delete

cluster_action_id The prefix which identifies the cluster

cluster_action_number The number which identifies the cluster

cluster_action_new Identifies whether the cluster is newly formed or existing

Return Code:

negative for error, 0 for success, 1 for end of results

ids_real_time_sync_start

Description:

Used to start a synchronizer transaction of the Real Time Synchronization server. This call works in a synchronous fashion returning only when the transaction has been processed. The record passed in must match the layout of the IDT. This call must be followed by one or more calls to `ids_real_time_sync_get` to retrieve results.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_sync_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
      <system>value</system>
      <IDT>value</IDT>
      <sequence_number>value</sequence_number>
    </ids_real_time_sync_start>
  </soap:Body>
</soap:Envelope>

```

```

    <operation>value</operation>
    <cluster_record>value</cluster_record>
    <cluster_record_size>50</cluster_record_size>
    <reference_size>256</reference_size>
  </ids_real_time_sync_start>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_real_time_sync_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <cluster_action_count>302</cluster_action_count>
      <reference>value</reference>
    </ids_real_time_sync_start_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

IDT is the name of the IDT associated with the update.

sequence_number is a string that specifies the order of synchronization. Must obey the rules for sequence numbers found in the OPERATIONS MANUAL.

operation The synchronizer operation being performed: A for add, D for delete or U for update

cluster_record The record to be updated. The record must use the same layout as the IDT.

cluster_action_count The number of clusters generated. For IDT/IDX only processing this parameter will always be 0. These can be returned with `ids_real_time_sync_get`

reference A reference string identifying the request. This must be passed to the `ids_real_time_sync_get` call when retrieving results.

Return Code:

negative for error

0 for success, and positive for warnings:

1-9 Reserved for future use

10 warning: Duplicate PK detected on add to IDT with `WARN_DUPLICATE_PK` sync option.

11 warning: Transaction was skipped.

12 warning: Transaction was reject because the sequence number was less than or equal to a previous transaction. The record was added to the reject table.

13 warning: Transaction became a No Op. e.g. Add followed by delete in the same commit cycle equates to do nothing.

14 warning: Force server shutdown in progress.

15 warning: Could not perform delete as the IDT record was not found. This will normally trigger error unless the appropriate option is set in the synchronization server configuration.

ids_scores_get

Description:

Retrieve an array of scores, one per match record. This API is used in conjunction with `ids_search_start` when candidate records to be matched are provided by the caller. The records and their scores may be retrieved either by repeatedly calling `ids_search_get` or by calling this function to retrieve all scores at once. A limit of 1024 scores may be returned in a single call. When using this function, please ensure that an `Accept` limit of 0 has been specified (so that all candidates are returned regardless of their score), and specify a `SORT=` parameter in the Search-Definition to ensure that the records remain in the same order as passed – otherwise they will be sorted by descending score. This is perhaps best achieved by inserting a record number in each row and sorting by that field. The size of the scores array must be equal to the number of input records to be matched, and may not exceed 1024.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_scores_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
      <scores_num>1</scores_num>
    </ids_scores_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_scores_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <scoresArray>
        <scores>301</scores>
      </scoresArray>
    </ids_scores_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

searchname is the name of the associated search

scores is an array of scores, one per candidate record

Return Code:

negative for error, 0 for success

ids_search_comment_get

Description:

Returns the user defined comment stored with the search.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_comment_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
      <comment_size>256</comment_size>
    </ids_search_comment_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_comment_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <comment>value</comment>
    </ids_search_comment_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

searchname the search to count.

comment is the area to copy the string containing the comment. This string will be null-terminated.

Return Code:

negative for error, 0 for success

ids_search_dedupe_start

Description:

Search for duplicate records in the IDT. Refer to the *Dup Finder* section in this manual for details.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_dedupe_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <search_width>value</search_width>
      <match_tolerance>value</match_tolerance>
      <parametersArray>
        <parameters>value</parameters>
      </parametersArray>
      <parameters_num>1</parameters_num>
      <parameters_size>50</parameters_size>
      <searchrec>value</searchrec>
      <searchrec_size>50</searchrec_size>
      <AnswersetName>value</AnswersetName>
      <flags>301</flags>
      <recid>303</recid>
      <recs>303</recs>
    </ids_search_dedupe_start>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_dedupe_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <searchrec>value</searchrec>
      <recid>303</recid>
      <recs>303</recs>
    </ids_search_dedupe_start_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

search is the name of the search that is to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates.

parameters not used.

searchrec is the IDT record used to search with. It is returned by the Search Server.

AnswersetName is used store the search results in an AnswerSet. The AnswerSet is used to identify the Search results in the table and is constructed by concatenating the `AnswersetName` parameter with the Search-Record-Id (10 bytes) . The maximum `AnswersetName` length is 22 characters. If you do not wish to store the search results in an AnswerSet, set `AnswersetName` to an empty String.

flags flags for specifying options. This field is a bit-field. Valid values are: 2 = return search record only. 4=remove search record from returned set.

recid the `recid` of the record to start a searching on. A value of 0 starts searching from the beginning of the IDT. The returned value is the `recid` of the next record to be searched.

recs the number of records in the search set.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set.

ids_search_fields_count

Description:

Gets the number of fields required to assemble the search record.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_fields_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
    </ids_search_fields_count>
  </soap:Body>
</soap:Envelope>
```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_fields_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <fc>302</fc>
    </ids_search_fields_count_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

searchname the search to count.

fc is the number of fields required to be filled in to assemble the search.

Return Code:

negative for error, 0 for success

ids_search_fields_get

Description:

Gets the number of fields required to assemble the search record.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_fields_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
      <fieldnames_num>1</fieldnames_num>
      <fieldnames_size>256</fieldnames_size>
    </ids_search_fields_get>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>

```

```

<ids_search_fields_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
  <response>0</response>
  <session>3436273</session>
  <fieldnamesArray>
    <fieldnames>value</fieldnames>
  </fieldnamesArray>
</ids_search_fields_get_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

searchname the search to count.

fieldnames is the array returned which will contain the name of the fields.

Return Code:

negative for error, 0 for success

ids_search_filter

Description:

Sets a dynamic SQL filter to be used by a search. Refere to the *SQL Filters* section of the DESIGNER MANUAL for details about SQL filters

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_filter
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <filter>value</filter>
    </ids_search_filter>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_filter_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>

```

```

    <session>3436273</session>
  </ids_search_filter_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the Search that will use the filter

filter is a string containing an SQL expression or values for substitution variables

Return Code:

negative for error, 0 for success

ids_search_finish

Description:

Release resources associated with `ids_search_start`.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_finish
      xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
    </ids_search_finish>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_finish_response
      xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_search_finish_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the search that was performed

Return Code:

negative for error, 0 for success

ids_search_get

Description:

Retrieve file records that are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
      <searchreturn_size>50</searchreturn_size>
      <sreps_num>1</sreps_num>
      <freps_num>1</freps_num>
    </ids_search_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <searchreturn>value</searchreturn>
      <score>302</score>
      <srepsArray>
        <sreps>301</sreps>
      </srepsArray>
      <frepsArray>
        <freps>301</freps>
      </frepsArray>
    </ids_search_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

searchname is the name the search to used by the call.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

sreps is an array of the ordinal values of the repeating fields in the search record that were used in the match.

Note: sreps and freps are only meaningful when using SSA-NAME3 v1, the SEQUENCES option has been set in the Search-Definition, and the search and file records contain repeating groups

freps is an array of the ordinal values of the repeating fields in the file record that were used in the match.

For example, a record structure with a repeating name (2 fields) and a repeating address (2 fields) , if the first name field in the search record matched the second name field in the file record while the first address field of the source matched the first address field of the file; the contents of these two arrays would be; sreps[0] = 0, sreps[1] = 0, freps[0] = 1, freps[1] = 0

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_complete

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information in the `info` field as long as the Search-Definition specifies the `SEQUENCES` option, and SSA-NAME3 v1 is used.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_get_complete
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <searchreturn_size>50</searchreturn_size>
      <info_size>50</info_size>
    </ids_search_get_complete>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_get_complete_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <searchreturn>value</searchreturn>
      <score>302</score>
      <info>value</info>
    </ids_search_get_complete_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

search is the name of the search which was performed.

searchreturn is an area into which a string from the set is copied.

score is the score calculated for the string.

info an encoded list of values used to determine which occurrence of a repeating field matched the value in the search record. The `info` field has a length of $4 \times (1 + 3 \times 100)$ bytes. It contains 4 groups, each one representing the result from one of the four possible scoring phases: Key-Pre-Score, Key-Score, Pre-Score and Score respectively. If a phase was used, its data starts with a 1, otherwise 0 if the phase was not used. The indicator is followed by 100 three-digit numbers, one each for each method in the scoring-scheme for this phase, up to a limit of 100 methods per scheme (scoring phase). The three-digit number is an index (base 1) representing the occurrence in the file record that was the best match for the data in the search record.

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_get_detail

Description:

Retrieve file records which are a good match for the search record specified in the `ids_search_start` or `ids_system_search_start` function. This function will return extended matching information, including the match decision and the file (IDT) record-ID of the matching records.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_get_detail
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
```

```

    <session>3436273</session>
    <search>value</search>
    <file_rec_size>50</file_rec_size>
    <decision_size>256</decision_size>
  </ids_search_get_detail>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_get_detail_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <file_rec>value</file_rec>
      <score>302</score>
      <decision>value</decision>
      <file_recid>302</file_recid>
    </ids_search_get_detail_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the Search which was performed.

file_rec is the matched File record.

score is the degree of similarity between the Search and File records (0-100).

decision is the match decision: A(ccept) or U(ndecided)

file_recid is the File Record-ID (corresponding to the RECID column from the IDT).

Return Code:

negative for error, 0 for success, and 1 for "end of set".

ids_search_IDT_get

Description:

Gets the name of the IDT associated with the search.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

```

```

<soap:Body>
  <ids_search_IDT_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
    <session>3436273</session>
    <searchname>value</searchname>
    <IDT_size>256</IDT_size>
  </ids_search_IDT_get>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_IDT_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <IDT>value</IDT>
    </ids_search_IDT_get_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

searchname is the name the search to used by the call.

IDT is the area into which the IDT name will be copied

Return Code:

negative for error, 0 for success

ids_search_layout

Description:

Get the names and descriptions of the fields in the current input or output view. If no view has been defined, the IDT layout will be returned.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_layout
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
    </ids_search_layout>
  </soap:Body>
</soap:Envelope>

```

```

    <viewType>value</viewType>
    <func>value</func>
    <names_num>1</names_num>
    <names_size>256</names_size>
    <lengths_num>1</lengths_num>
    <offsets_num>1</offsets_num>
    <repeats_num>1</repeats_num>
    <formats_num>1</formats_num>
    <formats_size>256</formats_size>
  </ids_search_layout>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_layout_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <namesArray>
        <names>value</names>
      </namesArray>
      <lengthsArray>
        <lengths>301</lengths>
      </lengthsArray>
      <offsetsArray>
        <offsets>301</offsets>
      </offsetsArray>
      <repeatsArray>
        <repeats>301</repeats>
      </repeatsArray>
      <formatsArray>
        <formats>value</formats>
      </formatsArray>
    </ids_search_layout_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the Search which was performed.

viewType the type of view: input or output.

func describes the order of fields.

names is the area into which an array containing the fieldnames will be copied.

lengths is the area into which an array containing the lengths of the fields will be copied.

offsets is the area into which an array containing the offsets of the fields will be copied.

repeats is the area into which an array containing the number of repeats in a field will be copied.

formats is the area into which an array containing the format a field will be copied. The format of a fields is a 50 character string in the following format:

Character 0:	Justification ('L'eft or 'R'ight)
Character 1:	Compression ('F'ixed, 'V'ariable or 'L'ong)
Characters 2 - 3:	Fill (2 characters containing the fill character in hexadecimal)
Character 4:	Fill type ('T'ext or 'B'inary)
Characters 5 - 6:	Base (2 characters containing the base in decimal)
Character 7:	Format ('T'ext, 'N'umeric, 'V'ariable or 'B'inary)
Character 8 - 9:	Reserved
Characters 10 - 11:	Binary key number (2 hexadecimal digits)
Character 12:	Character width ('W'ide, 'N'arrow)
Characters 13 - 50:	Reserved

Note: It is recommended that the `FORMATS_SIZE` constant be used to prevent errors from undersized strings.

Return Code:

negative for error, 0 for success

ids_search_profile_count

Description:

Count the number of profile entries available

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_profile_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_search_profile_count>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_profile_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_search_profile_count_response>
  </soap:Body>
</soap:Envelope>
```

```

        <count>302</count>
    </ids_search_profile_count_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

count Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_profile_get

Description:

Get the profiling times, with each field comprising of an id, a name and a time

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_profile_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <ids_num>1</ids_num>
      <names_num>1</names_num>
      <names_size>256</names_size>
      <times_num>1</times_num>
    </ids_search_profile_get>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_profile_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <idsArray>
        <ids>301</ids>
      </idsArray>
      <namesArray>
        <names>value</names>
      </namesArray>
    </ids_search_profile_get_response>
  </soap:Body>
</soap:Envelope>

```

```

    <timesArray>
      <times>301</times>
    </timesArray>
  </ids_search_profile_get_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

ids the numeric identifiers for each field, valid values are 0 The total time 1 The time to assemble the search record 2 The time to select the key field 3 The time to generate fuzzy ranges 4 The time to read candidates from the database 5 The time to score candidates 6 The time to sort candidates

names the names for each field

times the times for each field in microseconds Returns the number of elements

Return Code:

negative for error, 0 for success

ids_search_start

Description:

Performs a search using a pre-constructed search-record (searchrec). Alternatively you can supply fields (parameters) and have `ids_search_start` construct the record. The order of the field values in parameters should be the same as that returned by `ids_search_fields_get`. You can either search the database or search against a supplied list of records (records). There is a limit of 64K bytes of data that can be sent to the Server. If the supplied list of records is too large, split it into smaller groups and make multiple calls.

Note: The first search for a given System incurs an additional overhead to allocate database resources and access the Rulebase. Switching between searches on a particular connection is equivalent to starting a new search and therefore incurs some overhead. Applications requiring the best possible search performance should be designed to avoid switching between searches. The easiest way to do this is to use separate (dedicated) connections for each Search.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <search_width>value</search_width>
      <match_tolerance>value</match_tolerance>
      <parametersArray>

```

```

    <parameters>value</parameters>
  </parametersArray>
  <parameters_num>1</parameters_num>
  <parameters_size>50</parameters_size>
  <searchrec>value</searchrec>
  <searchrec_size>50</searchrec_size>
  <AnswersetName>value</AnswersetName>
  <recordsArray>
    <records>value</records>
  </recordsArray>
  <records_num>1</records_num>
  <records_size>50</records_size>
</ids_search_start>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <searchrec>value</searchrec>
      <recs>302</recs>
    </ids_search_start_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

AnswersetName is used store the search results in an `AnswerSet`. `AnswersetName` is used to identify the Search results in the table. The maximum `AnswersetName` length is 32 characters. If you do not wish to store the search results in an `AnswerSet`, set `AnswersetName` to an empty String.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

records contains a list of records to search on. If you wish to search on the database then specify this as containing 0 records.

Return Code:

negative for error, 0 for success, and 1 for truncation of the search set, 2 for exceeding SEARCH_LIMIT and 3 if the query timeout value set by `ids_set_timeout` or the environment variable `SSADB_QUERY_TIMEOUT` has been exceeded.

ids_search_start_via_parameters

Description:

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_start_via_parameters
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <search_width>value</search_width>
      <match_tolerance>value</match_tolerance>
      <parametersArray>
        <parameters>value</parameters>
      </parametersArray>
      <parameters_num>1</parameters_num>
      <parameters_size>50</parameters_size>
    </ids_search_start_via_parameters>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_start_via_parameters_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <datalen>302</datalen>
      <recs>302</recs>
    </ids_search_start_via_parameters_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

parameters contains the field values used to construct a search record (`searchrec`). The order of the field values must correspond to the order of fields returned by `ids_search_fields_get`. If insufficient fields are supplied, the remaining fields in the constructed search record will be blank filled.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_start_via_record

Description:

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_start_via_record
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <search_width>value</search_width>
      <match_tolerance>value</match_tolerance>
      <searchrec>value</searchrec>
      <searchrec_size>50</searchrec_size>
    </ids_search_start_via_record>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_start_via_record_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
```

```

    <datalen>302</datalen>
    <recs>302</recs>
  </ids_search_start_via_record_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the search to be performed.

search_width specifies either `Narrow`, `Typical` or `Exhaustive` to nominate how many candidates should be selected. If left blank, the `Search_Level` in the Controls will be used.

match_tolerance specifies either `Conservative`, `Typical` or `Loose` to nominate how aggressive the matching scheme should be in rejecting candidates. If left blank, the `Match_Level` in the Controls will be used.

searchrec is the record that we will search on (in IDT format, or the input view if specified). If `searchrec` is specified it will be used to search (provided no `parameters` are supplied). Alternatively if `parameters` are specified then the search will be on a record constructed on those parameters and returned to the user.

datalen will return the length of a record.

recs number of records that matched the search criteria. The count reflects the number of records prior to sorting the result set. If the sort uses the `Unique_Keys` option to remove duplicates, the resulting set may contain less than `recs` records. In this case, you must use the response code from `ids_search_get` to determine when the end of set has been reached.

Return Code:

negative for error, 0 for success

ids_search_tolerances_count

Description:

Returns the match tolerances count that have for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_tolerances_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
    </ids_search_tolerances_count>
  </soap:Body>
</soap:Envelope>

```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_tolerances_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <count>302</count>
    </ids_search_tolerances_count_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_tolerances_get

Description:

Returns the match tolerances that have been defined for the search. The match tolerance defines how aggressively the matching scheme should be in rejecting candidates.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_tolerances_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
      <tolerances_num>1</tolerances_num>
      <tolerances_size>256</tolerances_size>
    </ids_search_tolerances_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>
```

```

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_tolerances_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <tolerancesArray>
        <tolerances>value</tolerances>
      </tolerancesArray>
    </ids_search_tolerances_get_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

searchname the search to count.

tolerances is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_search_view_get

Description:

Returns the name of the current input or output view, together with information about the view: `view_field_count` is needed to dynamically allocate the arrays used for calls to `ids_search_layout` and `view_length` is used to dynamically allocate memory for input/output records.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_view_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <viewType>value</viewType>
      <viewName_size>256</viewName_size>
    </ids_search_view_get>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

```

```

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_view_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <viewName>value</viewName>
      <viewFieldCount>302</viewFieldCount>
      <viewRecLen>302</viewRecLen>
    </ids_search_view_get_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName is the name of the view to query

viewFieldCount the number of fields in the view

viewRecLen is the length of the view

Return Code:

negative for error, 0 for success

ids_search_view_set

Description:

Sets a view as the active input or output view

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_view_set
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <search>value</search>
      <viewType>value</viewType>
      <viewName>value</viewName>
    </ids_search_view_set>
  </soap:Body>
</soap:Envelope>

```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_view_set_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_search_view_set_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

search is the name of the Search

viewType the type of the view (input or output)

viewName the name of the view to use

Return Code:

negative for error, 0 for success

ids_search_widths_count

Description:

Returns the count of search widths that have been defined for the search. The search width defines how many items are selected by the search

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_widths_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
    </ids_search_widths_count>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
```

```

<ids_search_widths_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
  <response>0</response>
  <session>3436273</session>
  <count>302</count>
</ids_search_widths_count_response>
</soap:Body>
</soap:Envelope>

```

Parameters:

searchname is the name the search to used by the call.

count is the number of widths for the search by the call.

Return Code:

negative for error, 0 for success

ids_search_widths_get

Description:

Returns the search widths that have been defined for the search. The search width defines how many items are selected by the search

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_widths_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searchname>value</searchname>
      <widths_num>1</widths_num>
      <widths_size>256</widths_size>
    </ids_search_widths_get>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_search_widths_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>

```

```

        <widthsArray>
          <widths>value</widths>
        </widthsArray>
      </ids_search_widths_get_response>
    </soap:Body>
  </soap:Envelope>

```

Parameters:

searchname is the name the search to used by the call.

widths is the list of null terminated strings returned by the call.

Return Code:

negative for error, 0 for success

ids_server_version_get

Description:

Get the version information associated with the server.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_server_version_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <server_version_size>256</server_version_size>
    </ids_server_version_get>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_server_version_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <server_version>value</server_version>
    </ids_server_version_get_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

server_version is the area into which the string containing the version information will be copied.

Return Code:

negative for error, 0 for success

ids_session_close

Description:

Closes the session currently allocated. This will cause the databases kept open by the connection server to close and prevent the reuse of the session by subsequent calls by `ids_session_open`.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_session_close
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_session_close>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_session_close_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_session_close_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_session_open

Description:

Allocates resources for a socket. This is an API to the IDS Connection Server. The IDS Connection Server sits between the client and the IDS Search Server. A session prevents the IDS Search Server from reopening databases by keeping the databases open between connections. A timeout value can be specified when starting the IDS Connection Server. If the session has not been reused or closed before the timeout period it will be closed automatically.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_session_open
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>-1</session>
    </ids_session_open>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_session_open_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>303</session>
    </ids_session_open_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

session is the number of the session to open (-1 for a new session).

Return Code:

negative for error, 0 for success

ids_set_encoding

Description:

Informs the Search Server of the encoding used for fields of type 'W' (UNICODE data). ISS stores and retrieves W fields using an UTF-16 encoding. Search records should use this encoding for W columns if possible. If they do not, the caller must use this API to inform the Search Server of the alternate encoding used so that the Server can convert the data prior to using it. The W fields in each record of

the result set will be converted (if necessary) to the caller's encoding prior to return. If no encoding is specified, ISS assumes that the search data matches the file data and no conversion is performed. Refer to the *Globalization* section of the OPERATIONS manual for further details and restrictions.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_set_encoding
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <encoding>301</encoding>
    </ids_set_encoding>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_set_encoding_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_set_encoding_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

encoding 6=UTF-16/UCS-2 (LE), 7=UTF-16/UCS-2 (BE), 8=UTF-8, 4=UCS-4

Return Code:

negative for error, 0 for success

ids_set_timeout

Description:

Informs the Search Server of the timeout value in seconds The timeout value is used to set the SQL_ATTR_QUERY_TIMEOUT value. This should stop a long running search at the value of timeout although it depends on the configuration of the database server and database client as a default minimum timeout value may already be set (some database servers/clients may also have a periodic value for checking if the timeout value has been reached)

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_set_timeout
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <timeout>301</timeout>
    </ids_set_timeout>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_set_timeout_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_set_timeout_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

timeout timeout 0 is disable timeout, a positive value in seconds for the timeout

Return Code:

negative for error, 0 for success

ids_set_vpd_user

Description:

Provides the Search Server with information required to set a Virtual Private Database context. Refer to the DESIGNER MANUAL, *VPD* section for details about VPD.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_set_vpd_user
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
```

```

    <session>3436273</session>
    <vpd_user>value</vpd_user>
    <vpd_ctx>value</vpd_ctx>
  </ids_set_vpd_user>
</soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_set_vpd_user_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_set_vpd_user_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

vpd_user Connection string of the actual user

vpd_ctx the name of the PL/SQL context setting package

Return Code:

negative for error, 0 for success

ids_system_close

Description:

closes the system and frees any remaining resources.

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_close
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_system_close>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_close_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_system_close_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_system_idtname_count

Description:

Returns the number of active idt names. (i.e. those whose IDT has been loaded).

Input message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_idtname_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_system_idtname_count>
  </soap:Body>
</soap:Envelope>

```

Output message:

```

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_idtname_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <idtcount>302</idtcount>
    </ids_system_idtname_count_response>
  </soap:Body>
</soap:Envelope>

```

Parameters:

idtcount is the number of idtnames defined on the system

Return Code:

negative for error, 0 for success

ids_system_idtname_get

Description:

Get the names of all IDTs that are active. (i.e. those whose IDT has been loaded).

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_idtname_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <idtnames_num>1</idtnames_num>
      <idtnames_size>256</idtnames_size>
    </ids_system_idtname_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_idtname_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <idtnamesArray>
        <idtnames>value</idtnames>
      </idtnamesArray>
    </ids_system_idtname_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

idtnames is the area into which an array of the idtnames defined on the rulebase will be copied (the idtnames are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_system_notify

Description:

Notifies search server on a system.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_notify
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
      <sysname>value</sysname>
      <message>value</message>
    </ids_system_notify>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_notify_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_system_notify_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

rulebase is the name of the rulebase.

sysname is the name of the system

message is a message to be delivered

Return Code:

negative for error, 0 for success

ids_system_open

Description:

opens a system.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_open
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
      <system>value</system>
      <verbosity>value</verbosity>
      <Options>value</Options>
    </ids_system_open>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_open_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_system_open_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

Options LOGOUT filename for server output for this session.

LOGERR filename for server errors for this session.

LOGTEST filename for server search trace for this session.

SHOWALLSEARCHES modifies the behavior of

WORKDIR used to inform the search server as to which directory is to be used as the working directory for this session.

Return Code:

negative for error, 0 for success

ids_system_search_finish

Description:

Finishes the search and closes the system.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_search_finish
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_system_search_finish>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_search_finish_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
    </ids_system_search_finish_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

none

Return Code:

negative for error, 0 for success

ids_system_search_start

Description:

Opens a system and constructs and initialises a search using the fields passed to it in parameters. Refer to `ids_search_start` for a more detailed description of the parameters.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_search_start
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
      <system>value</system>
      <verbosity>value</verbosity>
      <options>value</options>
      <search>value</search>
      <parametersArray>
        <parameters>value</parameters>
      </parametersArray>
      <parameters_num>1</parameters_num>
      <parameters_size>50</parameters_size>
      <AnswersetName>value</AnswersetName>
    </ids_system_search_start>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_search_start_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <datalen>302</datalen>
      <recs>302</recs>
    </ids_system_search_start_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

rulebase is the name of the rulebase.

system The name of the system in the rulebase

verbosity specifies the verbosity level. See the *Verbosity* section for details.

options consists of one or more keywords and their respective values in the form KEYWORD1=VALUE1, KEYWORD2=VALUE2,

search is the name of the search in the system in the Rulebase that will be used.

parameters is the array which contains the field values to be used to construct the search.

AnswersetName is used store the search results in an AnswerSet. AnswersetName is used to identify the Search results in the table. The maximum AnswersetName length is 32 characters. If you do not wish to store the search results in an AnswerSet, set AnswersetName to an empty String.

dataLen will return the length of a record.

recs count of records that matched the search criteria

Return Code:

negative for error, 0 for success

ids_system_searches_count

Description:

Returns the number of runnable searches. (i.e. those whose IDX has been loaded). To return the number of defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_searches_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
    </ids_system_searches_count>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_searches_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <searchcount>302</searchcount>
    </ids_system_searches_count_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

searchcount is the number of searches defined on the system

Return Code:

negative for error, 0 for success

ids_system_searches_get

Description:

Get the names of all searches that are runnable. (i.e. those whose IDX has been loaded). To return the names of all defined searches, add the `SHOWALLSEARCHES` keyword to the option string of `ids_system_open`.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_searches_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <searches_num>1</searches_num>
      <searches_size>256</searches_size>
    </ids_system_searches_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_system_searches_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <searchesArray>
        <searches>value</searches>
      </searchesArray>
    </ids_system_searches_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

searches is the area into which an array of the searches defined on the rulebase will be copied (the searches are all null terminated strings).

Return Code:

negative for error, 0 for success

ids_systems_count

Description:

the number of systems in the rulebase.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_systems_count
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
    </ids_systems_count>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_systems_count_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <systemscount>302</systemscount>
    </ids_systems_count_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

rulebase is the name of the rulebase.

systemscount the number of systems in the rulebase.

Return Code:

negative for error, 0 for success

ids_systems_get

Description:

Get the names of all the systems defined in the rulebase.

Input message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_systems_get
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <session>3436273</session>
      <rulebase>value</rulebase>
      <systems_num>1</systems_num>
      <systems_size>256</systems_size>
    </ids_systems_get>
  </soap:Body>
</soap:Envelope>
```

Output message:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ids_systems_get_response
xmlns="http://www.identitysystems.com/xmlschema/iss-version-11">
      <response>0</response>
      <session>3436273</session>
      <systemsArray>
        <systems>value</systems>
      </systemsArray>
    </ids_systems_get_response>
  </soap:Body>
</soap:Envelope>
```

Parameters:

rulebase is the name of the rulebase.

systems is the area into which an array of the systems defined in the rulebase will be copied (the systems are all null terminated strings).

Return Code:

negative for error, 0 for success

Common Parameters

The following are the main input parameters used in the API function calls:

Verbosity

Some API functions must specify a verbosity level. The verbosity determines the amount of statistical information that will be written to the server log files. It is primarily used when tuning a search. Valid values are `-v [verbosity options]`.

Where *verbosity options* is a string consisting of one or more of the following characters:

0 (zero) turn off all verbosity

s search statistics

u database usage statistics

i info (used currently only by **updsync**)

Address Standardization

The Address Standardization APIs are used to parse addresses into their components fields and to validate them against postal reference databases. They can only be used successfully if the separately licensable *Address Standardization Module* has been installed.

This section of the documentation describes the semantics, process flow, parameter values / setting and returned data. The syntax of the API calls is described in the *Function Details* section of this manual.

Initialization

Each caller must initialize the Address Standardization sub-system by calling `ids_addr_init`. This API must be called prior to calling any other Address Standardization APIs.

Note: By default ASM uses AddressDoctor v4. To use ASM with AddressDoctor v5 call `ids_addr_set_option` API prior to calling `ids_addr_init`. Refer below table for `ids_addr_set_option`:

Option Name	Option Value	Meaning
LIBRARY_VERSION	V5	Use ASM with AddressDoctor v5.
	V4	Use ASM with AddressDoctor v4.

Character Sets and Countries

Initialization must be followed by a call to `ids_addr_set_attr`. This is used to define the character set of the input data. The parsed / validated address fields will be returned to the caller using this character set as well. A list of supported character sets appear below for ASM using AddressDoctor v4:

Character Set
UTF32BE
UTF32LE
UTF16BE
UTF16LE
UTF8
UCS4BE
UCS4LE
UCS2BE
UCS2LE
ISO8859_1

table continued on next page

table continued from previous page

Character Set
ISO8859_2
ISO8859_3
ISO8859_4
ISO8859_5
ISO8859_6
ISO8859_7
ISO8859_8
ISO8859_9
ISO8859_10
ISO8859_15
IBM437
IBM850
IBM852
IBM855
IBM857
IBM862
WIN1250
WIN1251
WIN1252
WIN1253
WIN1254
WIN1255
WIN1256
WIN1257
BIG5
JIS
GBK
UHC
ASCII
EBCDIC

Note: Character sets like UTF32BE, UTF32LE, UCS4BE and UCS4LE not supported by ASM using AddressDoctor v5.

As some of these character sets are multi-byte, `ids_addr_set_lines` and `ids_addr_get_field_idx` transfer address data using the Block data type (fixed length, not

nul-terminated pieces of memory).

The `ids_addr_set_attrib` API is also used to specify a default country. This value is used only when address parsing fails to find a valid country within the address. A list of valid country names appear below:

Country Name
Aruba
Afghanistan
Angola
Anguilla
Albania
Andorra
Netherlands Antilles
United Arab Emirates
Argentina
Armenia
American Samoa
Antarctica
French Southern Territories
Antigua and Barbuda
Australia
Austria
Azerbaijan
Burundi
Belgium
Benin
Burkina Faso
Bangladesh
Bulgaria
Bahrain
Bahamas
Bosnia and Herzegovina
Belarus
Belize
Bermuda
Bolivia
Brazil
Barbados
Brunei Darussalam
Bhutan
Bouvet Island
Botswana
Central African Republic
Canada
Cocos Islands
Switzerland
Chile
China
Cte d'Ivoire
Cameroon
Congo, The Democratic Republic of the
Congo

table continued on next page

table continued from previous page

Country Name
Cook Islands
Colombia
Comoros
Cape Verde
Costa Rica
Cuba
Christmas Island
Cayman Islands
Cyprus
Czech Republic
Germany
Djibouti
Dominica
Denmark
Dominican Republic
Algeria
Ecuador
Egypt
Eritrea
Western Sahara
Spain
Estonia
Ethiopia
Finland
Fiji
Falkland Islands
France
Faroe Islands
Micronesia, Federated States of
Gabon
United Kingdom
Georgia
Ghana
Gibraltar
Guinea
Guadeloupe
Gambia
GuineaBissau
Equatorial Guinea
Greece
Grenada
Greenland
Guatemala
French Guiana
Guam
Guyana
Hong Kong
Heard Island and McDonald Islands
Honduras
Croatia

table continued on next page

table continued from previous page

Country Name
Haiti
Hungary
Indonesia
India
British Indian Ocean Territory
Ireland
Iran, Islamic Republic of
Iraq
Iceland
Israel
Italy
Jamaica
Jordan
Japan
Kazakstan
Kenya
Kyrgyzstan
Cambodia
Kiribati
Saint Kitts and Nevis
Korea, Republic of
Kuwait
Lao People's Democratic Republic
Lebanon
Liberia
Libyan Arab Jamahiriya
Saint Lucia
Liechtenstein
Sri Lanka
Lesotho
Lithuania
Luxembourg
Latvia
Macau
Morocco
Monaco
Moldova, Republic of
Madagascar
Maldives
Mexico
Marshall Islands
Macedonia, The Former Yugoslav Republic of
Mali
Malta
Myanmar
Mongolia
Northern Mariana Islands
Mozambique
Mauritania
Montserrat

table continued on next page

table continued from previous page

Country Name
Martinique
Mauritius
Malawi
Malaysia
Mayotte
Namibia
New Caledonia
Niger
Norfolk Island
Nigeria
Nicaragua
Niue
Netherlands
Norway
Nepal
Nauru
New Zealand
Oman
Pakistan
Panama
Pitcairn
Peru
Philippines
Palau
Papua New Guinea
Poland
Puerto Rico
Korea, Democratic People's Republic of
Portugal
Paraguay
Palestinian Territory, Occupied
French Polynesia
Qatar
Runion
Romania
Russian Federation
Rwanda
Saudi Arabia
Sudan
Senegal
Singapore
South Georgia and the South Sandwich Islands
Saint Helena
Svalbard and Jan Mayen
Solomon Islands
Sierra Leone
El Salvador
San Marino
Somalia
Saint Pierre and Miquelon

table continued on next page

table continued from previous page

Country Name
So Tom and Prncipe
Suriname
Slovakia
Slovenia
Sweden
Swaziland
Seychelles
Syrian Arab Republic
Turks and Caicos Islands
Chad
Togo
Thailand
Tajikistan
Tokelau
Turkmenistan
Timor-Leste
Tonga
Trinidad and Tobago
Tunisia
Turkey
Tuvalu
Taiwan, Province of China
Tanzania, United Republic of
Uganda
Ukraine
United States Minor Outlying Islands
Uruguay
United States
Uzbekistan
Holy See
Saint Vincent and the Grenadines
Venezuela
Virgin Islands, British
Virgin Islands, U.S.
Viet Nam
Vanuatu
Wallis and Futuna
Samoa
Yemen
Serbia and Montenegro
South Africa
Zambia
Zimbabwe

The character set and default country are used for the life of the session or until they are changed.

Providing an Input Address

Address data can be provided in "10 line format" or "5 line format", by calling `ids_addr_set_lines`. When using "5 line format" (which is the most similar to an address as it would appear on an envelope),

the remaining lines should be passed as empty strings.

Alternatively, the address may be entered as individual fields, assuming that it has already been parsed. Use the `ids_addr_set_field_idx` API for this purpose.

When calling the CASS certified engine (`Val_Mode=Certify`), only 4 lines of input are accepted. Any additional lines will be ignored. The input lines must correspond to

- ♦ Organization,
- ♦ Delivery Address,
- ♦ Locality (City, State and Zip), and
- ♦ Country

When using "`Val_Mode=Complete`", an address must be entered as individual fields.

Parsing an Address

Parsing is the process of separating the address lines into separate address fields. It does not check whether or not the address fields constitute a valid postal address.

A call to `ids_addr_parse` returns a Long Array of field lengths. These correspond to the lengths of the parsed address fields, as per the table below. Fields with a length of zero simply mean that a particular address component was not present (for example, a middle name).

Field Index	Address Field
0	Organization
1	Department
2	Nobility (e.g. Lord)
3	Title (e.g. Mr)
4	First Name
5	Middle Name
6	Last Name
7	Function (e.g. Manager)
8	Building
9	Sub-Building
10	Street_1
11	Street_2
12	House Number
13	P.O. Box
14	Dependent Locality (e.g. URB, Colonia)
15	Locality (e.g. County)
16	Province (e.g. State)

table continued on next page

table continued from previous page

Field Index	Address Field
17	Zip Code
18	Country
19	Double Dependent Locality
20	Sorting Code (Mail Sort)
21	County Information
22	Zip+4
23	Geocoding Latitude
24	Geocoding Longitude
25	Geocoding Unit
26	Residue

For example, field length [13] represents the length of the "PO Box" component of the address.

Note: Some fields are not supported in AddressDoctor v4. ASM will not return values for these fields (i.e. geocoding related fields) when using AddressDoctor v4.

Validating an Address

Validating involves parsing the address into its component fields and checking those fields against a postal validation database. `ids_addr_validate` returns a `status` that indicates whether or not the address was valid, and if not, whether it could be corrected and its likelihood of being delivered successfully. Refer below table for validation status for ASM using AddressDoctor v4:

Status Code	Meaning
0	Address is correct
1	Address was corrected
2	Needs correction; deliverability high
3	Needs correction; deliverability fair
4	Needs correction; deliverability small
5	Country not recognized
6	No valid country database found
7	Country not unlocked
8	No validate called yet
9	Insufficient information
10	No suggestions
11	Suggestions incomplete

table continued on next page

table continued from previous page

Status Code	Meaning
12	Suggestions

Refer below table for validation status for ASM using AddressDoctor v5:

Status Code	Meaning
0	Verified - Input data correct - all elements were checked and input matched perfectly
1	Verified - Input data correct on input but some or all elements were standardised or input contains outdated names or exonyms
2	Verified - Input data correct but some elements could not be verified because of incomplete reference data
3	Verified - Input data correct but the user standardisation has deteriorated deliverability
4	Corrected - all elements have been checked
5	Corrected - but some elements could not be checked
6	Corrected - but delivery status unclear
7	Corrected - but delivery status unclear because user standardisation was wrong
8	Data could not be corrected completely, but is very likely to be deliverable - single match
9	Data could not be corrected completely, but is very likely to be deliverable - multiple matches
10	Data could not be corrected, but there is a slim chance that the address is deliverable
11	Data could not be corrected and is pretty unlikely to be delivered
12	FastCompletion Status - Suggestions are available - complete address
13	FastCompletion Status - Suggested address is complete but combined with elements from the input
14	FastCompletion Status - Suggested address is not complete
15	FastCompletion Status - Insufficient information provided to generate suggestions
16	Country recognized from ForceCountryISO3 Setting
17	Country recognized from DefaultCountryISO3 Setting
18	Country recognized from name without errors
19	Country recognized from name with errors
20	Country recognized from territory
21	Country recognized from province

table continued on next page

table continued from previous page

Status Code	Meaning
22	Country recognized from major town
23	Country recognized from format
24	Country recognized from script
25	Country not recognized - multiple matches
26	Country not recognized
27	Parsed perfectly
28	Parsed with multiple results
29	Parsed with Errors - Elements change position
30	Parse Error - Input Format Mismatch
31	Validation Error: No validation performed because country was not recognized
32	Validation Error: No validation performed because required reference database is not available
33	Validation Error: No validation performed because country could not be unlocked
34	Validation Error: No validation performed because reference database is corrupt or in wrong format
35	Validation Error: No validation performed because reference database is too old - please contact AddressDoctor to obtain updated reference data

The validation process may generate a number of suggested addresses when the input address is ambiguous. For example, the address

```
1520 Chestnut
Anytown AT 12345
```

may be ambiguous because both 1520 Chestnut Drive and 1520 Chestnut Court exist and without additional information, we cannot tell them apart. The number of suggestions returned is an output parameter from the validate call.

Retrieving Address Fields

After parsing or validation, individual address fields are available for collection as part of a "suggestion". Suggestion 0 always holds the parsed fields. Suggestions numbered 1 and above hold the validated address fields.

Individual fields are retrieved one by one, using `ids_addr_get_field_idx`, by nominating the suggestion index and field index. Since some fields may be missing, only those fields that have a non-zero length (as determined by calling `ids_addr_parse` or `ids_addr_get_field_info`) should be retrieved.

After validation, a field may be retrieved with `ids_addr_get_field`. This API is similar to `ids_addr_get_field_idx` but returns two extra codes that describe how the particular field

matched the validation data (`val_status`), and whether or not it was changed by the validation process (`val_mods`). Refer below table for `val_status` and `val_mods` for ASM using AddressDoctor v4:

val_status	Meaning
0	Empty
1	Not found
2	Not Checked (no reference data or no chance of success)
3	Matched with errors
4	Matched without errors

val_mods	Meaning
0	Empty
1	Not checked
2	Not checked but standardized
3	Checked and corrected (changed or inserted)
4	Validated, but changed (synonyms, old names)
5	Validated, but standardized
6	Validated and unchanged

Refer below table for `val_status` and `val_mods` for ASM using AddressDoctor v5:

val_status	Meaning
0	Empty
1	Not found
2	Not checked (no reference data)
3	Wrong - Set by validation only
4	Match with errors in this element
5	Match with changes
6	Match without errors

val_mods	Meaning
0	Empty
1	Not validated and not changed
2	Not validated bus standardized

table continued on next page

table continued from previous page

val_mods	Meaning
3	Validated but not changed due to invalid input
4	Validated but not changed due to lack of reference data
5	Validated but not changed due to multiple matches
6	Validated and changed by eliminating the input value
7	Validated and changed due to correction based on reference data
8	Validated and changed by adding value based on reference data
9	Validated, not changed, but delivery status not clear
12	Validated, verified but changed due to outdated name
13	Validated, verified but changed from exonym to official name
14	Validated, verified but changed due to standardization based on casing or language
15	Validated, verified and not changed due to perfect match

Setting Options

Some optional aspects of Address Standardization behavior may be set with the `ids_addr_set_option` API.

Option Name	Option Value	Meaning
Val_Mode	Correct	Correct the input address. Do not generate suggestions. Generally used in batch mode.
	Suggest	Generate suggestions (the default). Generally used by online applications where the operator can choose between a list of possibilities.
	Complete	Use an incomplete address (fragment) to quickly generate suggestions. Used for online "fast completion" style of applications.
	Certify	Use CASS certified validation rules defined by the USPS. Certify mode is only available for US addresses and requires additional database files to be installed in the DB directory.
Force_Country	True	Force the use of the Default Country
	False	Use Country detected from parsing the Address. This is the Default.
FORMAT_ZIP	True	To format the zip code in country specific format
	False	To unset the format of zip code in country specific format. This is the Default.

table continued on next page

table continued from previous page

Option Name	Option Value	Meaning
BASE_ZIP	True	To retrieve the zip code in Base format
	False	To unset the base format of zip code. This is the Default.

Getting Options

Some aspects of Address Standardization behavior may be obtained with the `ids_addr_get_option` API.

Option Name	Option Value	Meaning
VERSION	N/A	Gets the version number of the validation database

Sample Code

Sample code that demonstrates the API calls is in the `samples` directory. C Code for example can be found here **`ssaas/ad/samples/addrstd.c`**.

This code is compiled and linked in the same way as the other sample programs found in the Client and Developer Components install package in **`samples/programs`**.

It only requires one command line parameter to run: the `host:port` of the Search Server. For example

```
addrstd -h%SSA_SEPORT%
```

The sample code makes use of the Swiss validation database. This should be installed prior to use. See the ***Validation Database Files*** section for details.

The sample code by default makes use of ASM with AddressDoctor v5. To use ASM with AddressDoctor v4 run sample code with `-v4` command line option. For example

```
addrstd -h%SSA_SEPORT% -v4
```

Validation Database Files

The validation process makes use of postal validation databases stored in a sub-directory of the IIR Server's installation named **`ssaas/ad/ad/db`** for ASM using AddressDoctor v4 and **`ssaas/ad5/ad/db`** for ASM using AddressDoctor v5.

The Address Standardization Module installer does not install any validation databases. They are ordered separately by contacting Informatica LLC. When you receive these files (named **`*.MD`**), you

must copy them into the directory specified above. You will also receive a file named **key** that contains an unlock code for your specific databases. This file must be copied to the same directory. Also in case of ASM using AddressDoctor v5 to support geocoding, unlock code for geocoding should be present in file named **key.geo**. This file must be copied to the same directory where **key** is copied.

ASM configuration for AddressDoctor v5

The values of unlock key, number of threads, maximum number of address object to be used for Address Standardization are determined from the values in the file **ssaasmv5.xml**.

Note: The format of the ASM ADv5 configuration XML should be in UTF-8.

Address Standardization XML configuration file samples:

The simplest XML configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ASM_ADv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
</ASM_ADv5_Config>
```

Note: MAX_THREAD value should not be set to a larger value than the number of cores/CPUs. Its recommended to set the value of MAX_ADOBJECTS as twice the number of threads set using MAX_THREAD. Also these options should be set before starting IIR servers.

XML Configuration with multiple license key:

```
<?xml version="1.0" encoding="UTF-8"?>
<ASM_ADv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 2**</UNLOCK_CODE>
    ...
    <UNLOCK_CODE>**Placeholder for UnlockCode N**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
</ASM_ADv5_Config>
```

XML configuration with preload options:

```
<?xml version="1.0" encoding="UTF-8"?>
<ASM_ADv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
  <PRELOAD_COUNTRIES>
    <PRELOAD TYPE="FULL">Switzerland</PRELOAD>
    <PRELOAD TYPE="PARTIAL" VALMODE="CORRECT">Canada</PRELOAD>
    <PRELOAD VALMODE="CERTIFY">Australia</PRELOAD>
  </PRELOAD_COUNTRIES>
</ASM_ADv5_Config>
```

Note: Refer to section *Character Sets and Countries* chapter for valid country name for preload.

XML configuration with enrichment options:

```
<?xml version="1.0" encoding="UTF-8"?>
<ASM_Adv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
  <PRELOAD_COUNTRIES>
    <PRELOAD TYPE="PARTIAL" VALMODE="CERTIFY">Australia</PRELOAD>
  </PRELOAD_COUNTRIES>
  <PROCESS_OPTION>
    <ENRICHMENT_OPTION>EnrichmentGeoCoding</ENRICHMENT_OPTION>
    <ENRICHMENT_OPTION>EnrichmentCASS</ENRICHMENT_OPTION>
  </PROCESS_OPTION>
</ASM_Adv5_Config>
```

Note: To get CASS, GeoCoding, AMAS, SERP enrichment fields we need to specify enrichment options. **EnrichmentGeoCoding, EnrichmentCASS, EnrichmentAMAS, EnrichmentSERP, EnrichmentSNA** etc. are the possible values for enrichment option.

XML configuration with input options:

```
<?xml version="1.0" encoding="UTF-8"?>
<ASM_Adv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
  <PRELOAD_COUNTRIES>
    <PRELOAD TYPE="PARTIAL" VALMODE="CERTIFY">Australia</PRELOAD>
  </PRELOAD_COUNTRIES>
  <PROCESS_OPTION>
    <ENRICHMENT_OPTION>EnrichmentGeoCoding</ENRICHMENT_OPTION>
    <ENRICHMENT_OPTION>EnrichmentCASS</ENRICHMENT_OPTION>
  </PROCESS_OPTION>
  <INPUT_OPTION>
    <INPUT_PARAMETER TYPE="FormatDelimiter">COMMA</INPUT_PARAMETER>
  </INPUT_OPTION>
</ASM_Adv5_Config>
```

Note: FormatDelimiter is from a choice of (default is CRLF): CRLF, LF, CR, SEMICOLON, COMMA, TAB, PIPE or SPACE.

XML configuration with result options:

```
<?xml version="1.0" encoding="UTF-8"?>
<ASM_Adv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
  <PRELOAD_COUNTRIES>
    <PRELOAD TYPE="PARTIAL" VALMODE="CERTIFY">Australia</PRELOAD>
  </PRELOAD_COUNTRIES>
  <PROCESS_OPTION>
    <ENRICHMENT_OPTION>EnrichmentGeoCoding</ENRICHMENT_OPTION>
    <ENRICHMENT_OPTION>EnrichmentCASS</ENRICHMENT_OPTION>
  </PROCESS_OPTION>
```

```

<INPUT_OPTION>
  <INPUT_PARAMETER TYPE="FormatDelimiter">COMMA</INPUT_PARAMETER>
</INPUT_OPTION>
<RESULT_OPTION>
  <RESULT_PARAMETER TYPE="AddressElements">DETAILED</RESULT_PARAMETER>
</RESULT_OPTION>
</ASM_Adv5_Config>

```

Note: AddressElements is from a choice of (default is STANDARD): NONE, STANDARD or DETAILED. Setting AddressElements to DETAILED displays all residue elements.

XML configuration with postal reference database path:

```

<?xml version="1.0" encoding="UTF-8"?>
<ASM_Adv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <POSTAL_DB_PATH>**Placeholder for postal database path**</POSTAL_DB_PATH>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
  <PRELOAD_COUNTRIES>
    <PRELOAD TYPE="PARTIAL" VALMODE="CERTIFY">Australia</PRELOAD>
  </PRELOAD_COUNTRIES>
  <PROCESS_OPTION>
    <ENRICHMENT_OPTION>EnrichmentGeoCoding</ENRICHMENT_OPTION>
    <ENRICHMENT_OPTION>EnrichmentCASS</ENRICHMENT_OPTION>
  </PROCESS_OPTION>
  <INPUT_OPTION>
    <INPUT_PARAMETER TYPE="FormatDelimiter">COMMA</INPUT_PARAMETER>
  </INPUT_OPTION>
</ASM_Adv5_Config>

```

XML configuration with postal reference database path for preload:

```

<?xml version="1.0" encoding="UTF-8"?>
<ASM_Adv5_Config>
  <MAX_THREAD>1</MAX_THREAD>
  <MAX_ADOBJECTS>2</MAX_ADOBJECTS>
  <POSTAL_DB_PATH>**Placeholder for postal database path**</POSTAL_DB_PATH>
  <AD5_UNLOCK_CODE>
    <UNLOCK_CODE>**Placeholder for UnlockCode 1**</UNLOCK_CODE>
  </AD5_UNLOCK_CODE>
  <PRELOAD_COUNTRIES>
    <PRELOAD TYPE="FULL"
POSTAL_DB_PATH="**Placeholder for postal database path**">United States</PRELOAD>
    <PRELOAD VALMODE="CERTIFY">Germany</PRELOAD>
  </PRELOAD_COUNTRIES>
  <PROCESS_OPTION>
    <ENRICHMENT_OPTION>EnrichmentGeoCoding</ENRICHMENT_OPTION>
    <ENRICHMENT_OPTION>EnrichmentCASS</ENRICHMENT_OPTION>
  </PROCESS_OPTION>
  <INPUT_OPTION>
    <INPUT_PARAMETER TYPE="FormatDelimiter">COMMA</INPUT_PARAMETER>
  </INPUT_OPTION>
</ASM_Adv5_Config>

```

Batch Test Utility

Provided with the Address Standardization Module is the batch utility **asmiss**. This program utilizes all of the API functions. It takes address from an input file and can perform both parsing and validation. It is ideal for verifying programs that utilize the Address Standardization Module API or for batch processing a number of addresses.

```
%SSABIN%\asmiss InputFile -hHostName:PortNumber [Options]
```

Where

InputFile The input file will consist of addresses in a ten line format. Addresses with less than 10 lines must be terminated with a comment line. This is a line beginning with the comment string which defaults to the '#' character.

-hHostName:PortNumber Search Server Host name and Port Number.

-a Prints the suggested address label.

-A Force the use of archive tables.

-b Force the use of ASM using AddressDoctor v5 (default v4).

-cCharSet The Character set to use. The default is WIN1250.

-dDefaultCountry The Country to use when parsing can not determine a country from the address.

-f Force the use of the Default Country.

-gFileNameCASS summary report file-name.

-i Individual field-level input. When specified, each input address consists of n lines, where n is the maximum number of Field-Index values supported, as documented in the *Parsing An Address* section. Each field value must be prefixed by 3 bytes (which are ignored). This reserves space for a 2 digit field-index plus a space, used for documentation purposes. For example, a valid input file may look like this:

```
00
01
02
03 Mr.
04 John
05 Peter
06 Smith
07
08
09
11
12 100 Apt 23A
13
14
15
16 NY
17 10023
18
\#
```

-l Remove line separator from input.

-Lpreload_country The Country to use during preloading of database into memory.

-mValMode The Mode to use for validation purposes valid values are Suggest, Correct, Complete and Certify. The default value is Suggest.

-o PO Box complete flag.

-P Force the use of partially fielded. This should be accompanied with Individual field-level input.

-rString Set the comment character/string. This defaults to '#'.

- S** Prints the address match score.
- Tpreload_type** Set the preload type This defaults to 'NOPRELOAD'.
- v** Performs Validation. The default is to parse only.
- xml** Generate CASS 3553 summary report in xml format..
- y** Set preferred language type.
- z** Prints the Validation database version.
- ZzipFormat** Set the format of the output zip code. The valid values are `BASE_ZIP` and `FORMAT_ZIP`.

AnswerSet

AnswerSet provides the ability to store the search results in a database table. Some programming environments may not have the ability to retrieve search results, so it may be more convenient to initiate a search using the APIs and read the search results from a database table using SQL.

Table Name and Index

An AnswerSet table is created automatically the first time it is required. Only one AnswerSet table is created per IDT and is shared by all users of that IDT.

The table is named `IDS_SystemQualifier_AS_IDTname` where

SystemQualifier is the qualifier used in the Database Name (eg the 01 in "odb:01:scott/tiger@oracle")

IDTname is the IDT-NAME= parameter from the IDX-Definition. The length of the table name is limited by the target database.

An AnswerSet index is also created. Its name has the following format:

`IDSX_ SystemQualifier_AS_ IDTname`

Storage options (extents, tablespace, etc) for an AnswerSet are defined using the DATABASE-OPTIONS= parameter in the System-Definition.

Table Layout

An AnswerSet Table contains the following columns:

- ◆ Set-Id CHAR(32)
- ◆ ScoreNUMBER (3,0)
- ◆ File Record-IdNUMBER (10)
- ◆ the columns from the IDT

The *Set-Id* is used to identify a result set. The caller of the API specifies this value (maximum of 32 bytes). The *Score* represents the score (0-100) of how well the search record matches the file record. The *File Record-Id* uniquely identifies the file record and corresponds to the *RecId* column in the IDT.

Adding Rows

The AnswerSet table is populated with rows as a result of calling `ids_search_start` or `ids_system_search_start` and specifying a Set-Id.

If a Set-Id is already in use, the new search results will overwrite the existing set.

Clearing the Table

All AnswerSets can be cleared by dropping the AnswerSet table and AnswerSet index. This should only be done when the Search Server is down. The table and index will be recreated automatically when required.

Oracle / MS-SQL Server: The table and index can be truncated to clear all rows.

Relate & DupFinder Set-Ids

Relate and DupFinder can write their output to an AnswerSet.

For Relate, Set-Ids are created by concatenating the Set-Id prefix (`-a` parameter) to the record number from the input file.

For DupFinder, Set-Ids are created by concatenating the Set-Id prefix (`-a` parameter) to the record-id of the search record.

Oracle RAC

Customers running with Oracle's Real Application Clusters (RAC) may need to set the initialization parameter `MAX_COMMIT_PROPAGATION_DELAY` to 0. This will prevent a delay in synchronizing AnswerSets committed on one instance and read from another. Refer to MetaLink DocId 259454.1 for details.

DupFinder

The DupFinder is a facility for detecting duplicate IDT records. It is accessed via the search API. Instead of passing a search record to search with, it is called with special parameters that tell it to acquire search records from the IDT.

It uses the same calling protocol as a normal search; start a search with `ids_search_dedupe_start`, read set members with `ids_search_get`, close the set with `ids_search_finish`.

Each set contains the duplicate records including the search record, or optionally, the search record only, or all records without the search record.

Normally DupFinder only returns to the caller when it finds a duplicate or has processed all the IDT records. This saves on network traffic. However, you can control how many IDT records are searched before DupFinder returns. You may wish it to return after processing a fixed number of records (`DEDUPE-PROGRESS=` in the Multi-Search Definition) if you do not expect many dupes and/or want to search a range of record-ids. The search returns a flag indicating the number of records found, and the last Record-Id it searched.

If it found duplicates and you specified an AnswerSet id (`SetId`) within the `ids_search_dedupe_start`, use the `SetId` to retrieve the set members from the AnswerSet table. If you did not specify an AnswerSet id (`SetId`) you can retrieve the set members using repeated calls to `ids_search_get`. Close the set using `ids_search_finish`. Issue a new search to start the next search.

Each `ids_search_dedupe_start` call passes a parameter named `recid` that sets the starting Record-Id for the search. Set it to 0 to start a new sweep of the IDT. On return `ids_search_dedupe_start` returns the `recid` of the source record that has a duplicate. Simply pass the same value to the next `ids_search_dedupe_start` call to find the next set of dupes. If the `recid` is 0, it means it has reached the end of the IDT and there are no more records left to process.

`ids_search_dedupe_start` will return an empty set (`recs=0`) when the search record matched no IDX records, or if `DEDUPE-PROGRESS=` was specified and the max was reached. If `DEDUPE-PROGRESS=` was not specified, the situation indicates an integrity error, or a lack of synchronization between the IDT and IDX. In this situation, it also writes a warning message on the search server's log.

Here is some java sample code illustrating the use of the functions:

```
for (;;) {
    rc = ssaid.ids_search_dedupe_start (
        search,          /* IN: Search name */
        "Typical",       /* IN: Search Width */
        "Typical",       /* IN: Match Tolerance */
        parameters,      /* IN: Search values - not used */
        searchRec,       /* IN: search record */
        searchRec.length,
        setId,
        flags,
        recId,
        recsFound);      /* OUT: Number of records found */

    System.out.println (prog + "> [" + currentSearch +
        "] Search_Record = '" +
        new String (searchRec) + "'");
    System.out.println (prog + "> [" + currentSearch +
        "] records found=" + recsFound[0]);
}
```

```

        System.out.println (prog + "> [" + currentSearch +
            "]" recId = " + recId[0]);

        if (1 == rc) {
            System.out.println (prog + "> Dedupe truncated.");
        }

        if (0 == recId[0]) {
            System.out.println (prog + "> Dedupe completed.");
            break;
        }

        if (0 == recsFound[0]) {
            System.out.println (prog +
                "> Dedupe progress notification.");
            continue;
        }

        while (0 == ssaid.ids_search_get (
            search,                /* IN: Search name */
            rec,                   /* OUT: Record */
            searchRec.length,
            score,                 /* OUT: Score */
            sreps, 0,              /* OUT: Sreps - not used */
            freps, 0)) {          /* OUT: Freps - not used */

            System.out.println (prog + "> [" + currentSearch +
                "]" IDT_Record = '" +
                new String (rec) + "'");
            System.out.println (prog + "> [" + currentSearch +
                "]" Score = " + score [0]);
        }

        ssaid.ids_search_finish (search);
        ++currentSearch;
    }
}

```

AnswerSet

DupFinder results can be stored in an AnswerSet . In order to identify the search record, the Set-Id of the AnswerSet is created by concatenating the Set-Id provided by the caller to the 10 byte Search Record-Id from the IDT.

Therefore, when adding DupFinder results to an AnswerSet the Set-Id parameter is limited to 22 bytes.

MVS: AnswerSets created by a DupFinder must be placed in a separate tablespace using the DATABASE-OPTIONS= clause.

Match Explain API

Match Explain Record Formats

Match Explain records are described by the header `ssaexpl.h`. There are 6 different types of record. Each record start with a series of 9 ascii digits which are explained as:

1. 3 digits describing the type of record
2. 3 digits showing the parent sequence number of the record (0 = no parent) - which is useful for implementing tree displays
3. 3 digits showing the sequence number of the record

The types of record and what follows is shown below:

1. Type 000 The base record (there will be one of these for each phase performed, and parent sequence number will be 0) gives the overall result which is already available to the user. The record header is followed by:

- (a) A 3 digit number for score
- (b) A single character for decision (A = Accepted, R = Rejected, U = Undecided).
- (c) A 3 digit match explain API version number (the current version is 001) (new for 9.5.3)
- (d) A single character denoting the phase the summary describes (L)WM or Name(3) (new for 9.5.3)
- (e) A single character showing if LWM was performed (Y/N) (new for 9.5.3)
- (f) A single character showing if Name3 was performed (Y/N) (new for 9.5.3)

2. Type 001 This level of record displays operators and brackets. The header is followed by a 2 digit number indicating which operator applies. The meaning of codes are:

01	AND
02	OR
03	Open Bracket
04	Close Bracket
05	NOT

3. Type 002 This level of record give the summary for an SSA-NAME3 Purpose. The header is followed by 76 bytes for Purpose Name, 3 bytes for score, 1 byte for decision, 32 bytes for match tolerance, a 3 digit number for each of accept and reject limits and a single byte flag (Y/N) indicating if an early exit was taken.

4. Type 003 Type 3 records indicate the grouping of fields in a Purpose – currently there are best and required groups. A single byte (B/R) indicating which type of group follows the record header. The remainder of the record is dependent on the type of group. 'Best' groups are following by a 3 digit number which indicates how many bytes are used for the label of the group. 'Required' groups are followed by a 2 digit number indicating how many group members must be found, a 2 digit number showing drop and a 2 digit number showing slide, followed by a 3 digit number indicating length of the final label field. Drop indicates the maximum field score available and Slide indicates the score penalty applied if not all group members are present.

5. Type 004 This record describes field (method) results. The header is followed by 32 bytes indicating field name, a 3 digit score, a 4 digit weight (scale factor of 100 has been applied), a 3 digit original weight (at original scale), a weight flag (W = weight, A = adjusted) byte, a contributed flag byte (Y/N), an optional flag byte (Y/N), a 3 digit field contribution; a repeating field flag (Y/N), and a 2 byte index for each of search and file fields used (will be 0 if not used)

6. `Type 005` This record simply shows the field data used in the match call. The header is followed by a single byte indicating if (F)ile or (S)earch data, then a 3 digit number indicating the length of the field data which completes the record.

Match Explain Scores

As described in the previous section, `Type 004` records contain the scores achieved by individual fields. Note that these individual scores may not add up to the score returned for the whole Purpose. This is due to the fact that these field scores have already been rounded up or down before being returned to the user.

Index

AnswerSet, 19, 499, 501, 502

C, 30

C#, 135

CICS, 184

Cobol, 184

Compile, 22, 184, 493

HTTP, 391

Java, 238

Perl, 292

SOAP, 392

Unicode, 394

verbosity, 479

Visual Basic .NET, 341

WSDL, 391

XML, 391

XML Search Server, 391