

IBM Cognos Analytic Server
Version 10.1.0

API Guide



Note

Before using this information and the product it supports, read the information in "Notices" on page 415.

Product Information

This document applies to IBM Cognos Express Version 10.1.0 and may also apply to subsequent releases. To check for newer versions of this document, visit the IBM Cognos Information Centers (<http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp>).

Microsoft product screen shot(s) used with permission from Microsoft.

Licensed Materials - Property of IBM

© **Copyright IBM Corporation 2007, 2012.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Introduction	xv
Chapter 1. The IBM Cognos TM1 programming environment	1
Location of the IBM Cognos TM1 API DLLs	1
Setting a path to the DLLs	1
Supported compilers	2
Servers	2
Local servers	2
Remote servers	2
The admin host and admin server	3
The role of an IBM Cognos TM1 server	3
Multitasking and symmetric multiprocessor support	4
IBM Cognos TM1 server performance	4
Sample code for the IBM Cognos TM1 API	4
Chapter 2. IBM Cognos TM1 programming concepts	5
System functions	5
Connecting to the API	5
Disconnecting from the API	8
Managing server connections	8
Setting the admin host server	8
Progress callback functions	8
Requesting the version of the system	9
Thread safety	9
Values	9
Handles	9
Simple IBM Cognos TM1 data types	10
Value handling functions	10
String handling	11
Object handling	12
Array handling	12
Updating value capsules	12
Pending values	13
Managing value pools	13
Object attributes	14
Security	14
Security levels	14
Groups	15
Clients	15
Assigning clients to groups	16
Assigning rights to objects and groups	16
Meta security	16
Managing locks and reservations	16
Determining access rights for a client	16
Error handling	17
Examining error values	17
Error codes	17
API error codes for data reservations	17
Backup and recovery	18
The transaction log file	18
Treatment of time	19
Naming conventions	19
Log file structure	19
Automatic recovery	19
Accessing the log files	20

Logging changes to dimensions.	20
Creating an IBM Cognos TM1 project in C or C++	20
Generating a console project.	21
Setting paths	21
Adding test code	22
Building and running the program	22
Creating an IBM Cognos TM1 project in Microsoft Visual Basic	22
Adding code to your Microsoft Visual Basic Project	23
Logging in to an IBM Cognos TM1 Server	24
Logging out of IBM Cognos TM1	25

Chapter 3. IBM Cognos TM1 objects 27

IBM Cognos TM1 objects overview	27
Object handles	27
Object properties	28
Lists of objects	29
Creating and registering objects	30
Registration functions	31
Public, private, and unregistered objects	31
Accessing parent objects - security.	32
Object names	32
Loading objects	33
Deleting objects	33
Saving objects to disk	34
Copying objects	34
Server object	34
Server properties	34
Server functions	35
Dimension and element objects.	36
Parent object	36
Child objects	36
Registration	36
Creating a dimension	36
Updating a dimension.	37
Deleting a dimension	37
Elements	37
Dimension properties	38
Dimension functions	38
Cube objects	39
Parent object	39
Child objects	39
Registration	39
Creating a cube	40
Retrieving and updating cube cells	40
Deleting a cube	40
Cube properties	40
Cube functions	41
Cube perspectives	41
Rule objects	42
Rule properties	42
Rule functions	42
Creating a new rule	43
Updating an existing rule.	43
Subset objects.	43
Parent object	43
Registration	44
Creating a subset	44
Deleting a subset	44
Updating a subset	44
Subset object properties	44
Subset functions.	45

Subset element display functions	46
View objects	48
Parent object	48
Registration	48
Creating a view	48
Deleting a view	48
View object properties	49
View functions	50
BLOB objects	51
Property object	51
Child object	51
Registration	51
Creating a BLOB	51
Updating a BLOB	51
Deleting a BLOB.	51
File storage	52
BLOB object properties	52
Blob functions	52

Chapter 4. IBM Cognos TM1 functions for C 53

Function types and naming conventions	53
Configuring the IBM Cognos TM1 C API to use SSL.	53
TM1SystemSetAdminSSLCertAuthority	53
TM1SystemGetAdminSSLCertAuthority	53
TM1SystemSetAdminSSLCertRevList	54
TM1SystemGetAdminSSLCertRevList.	54
TM1SystemSetAdminSSLCertID	54
TM1SystemGetAdminSSLCertID	54
TM1SystemSetExportAdminSvrSSLCertFlag	54
TM1SystemSetAdminSvrExportKeyID	54
TM1APIInitialize	55
TM1APIFinalize	55
TM1BlobClose	56
TM1BlobCreate	56
TM1BlobGet	57
TM1BlobOpen	57
TM1BlobPut	58
TM1CancelClientJob	59
TM1ChangeSetBegin	59
TM1ChangeSetEnd	60
TM1ChangeSetUndo	61
TM1ChoreExecute	61
TM1ClientAdd	62
TM1ClientGroupAssign	62
TM1ClientGroupIsAssigned	63
TM1ClientGroupRemove	64
TM1ClientHasHolds	64
TM1ClientPasswordAssign	65
TM1ConnectionCheck	66
TM1ConnectionCreate	66
TM1ConnectionDelete	67
TM1ConnectionSynchronize	68
TM1CubeCellDrillListGet	68
TM1CubeCellDrillObjectBuild	69
TM1CubeCellPickListGet	70
TM1CubeCellsPickListGet	71
TM1CubeCellPickListExists	72
TM1CubeCellSpreadViewArray.	72
TM1CubeCellSpread	77
TM1CubeCellSpreadStatusGet	81
TM1CubeCellValueGet.	83

TM1CubeCellsValueGet	84
TM1CubeCellValueSet	84
TM1CubeCellsValueSet	85
TM1CubeCreate	86
TM1CubeDimensionListGet	86
TM1CubeListByNamesGet	87
TM1CubeListGet	88
TM1CubePerspectiveCreate	88
TM1CubePerspectiveDestroy	90
TM1CubeShowsNulls	90
TM1CubeTimeLastInvalidated	91
TM1DataReservationAcquire	91
TM1DataReservationGetAll	92
TM1DataReservationGetConflicts	93
TM1DataReservationRelease	94
TM1DataReservationReleaseAll	95
TM1DataReservationValidate	95
TM1DimensionAttributesGet	96
TM1DimensionCreateEmpty	96
TM1DimensionCheck	97
TM1DimensionElementComponentAdd	98
TM1DimensionElementComponentDelete	99
TM1DimensionElementComponentWeightGet	100
TM1DimensionElementDelete	101
TM1DimensionElementInsert	101
TM1DimensionElementListByIndexGet	107
TM1DimensionElementListByNamesGet	107
TM1DimensionUpdate	108
TM1ElementComponentsGet	109
TM1GetSubsetByHandle	110
TM1GetViewByName	110
TM1GetViewByHandle	113
TM1GroupAdd	113
TM1ObjectAttributeDelete	114
TM1ObjectAttributeInsert	115
TM1ObjectAttributeValueGet	116
TM1ObjectAttributeValueSet	116
TM1ObjectCopy	117
TM1ObjectDelete	118
TM1ObjectDestroy	118
TM1ObjectDuplicate	119
TM1ObjectFileDelete	120
TM1ObjectFileLoad	120
TM1ObjectFileSave	121
TM1ObjectListCountGet	122
TM1ObjectListHandleByIndexGet	122
TM1ObjectListHandleByNameGet	123
TM1ObjectPrivateDelete	124
TM1ObjectPrivateListCountGet	124
TM1ObjectPrivateListHandleByIndexGet	125
TM1ObjectPrivateListHandleByNameGet	126
TM1ObjectPrivatePublish	127
TM1ObjectPrivateRegister	127
TM1ObjectPropertyGet	128
TM1ObjectPropertySet	129
TM1ObjectRegister	130
TM1ObjectReplicate	130
TM1ObjectSecurityLock	131
TM1ObjectSecurityRelease	132
TM1ObjectSecurityReserve	132
TM1ObjectSecurityRightGet	133

TM1ObjectSecurityRightSet	134
TM1ObjectSecurityUnLock	135
TM1ProcessExecute	135
TM1ProcessExecuteEx	137
TM1ProcessExecuteSQLQuery	138
TM1ProcessVariableNamelsValid	138
TM1RuleAttach.	139
TM1RuleCheck	139
TM1RuleCreateEmpty	140
TM1RuleDetach	141
TM1RuleLineGet	141
TM1RuleLineInsert	142
TM1ServerBatchUpdateFinish	143
TM1ServerBatchUpdateIsActive	143
TM1ServerBatchUpdateStart	144
TM1ServerDimensionListGet	145
TM1ServerDimensionListByNamesGet	145
TM1ServerDisableBulkLoadMode	146
TM1ServerEnableBulkLoadMode	146
TM1ServerLogClose	147
TM1ServerLogNext	147
TM1ServerLogOpen	148
TM1ServerOpenSQLQuery	149
TM1ServerPasswordChange	150
TM1ServerSandboxesDelete	151
TM1ServerSecurityRefresh	154
TM1SQLTableGetNextRows	155
TM1SubsetAll	155
TM1SubsetCreateByExpression	156
TM1SubsetCreateEmpty	157
TM1SubsetElementDisplay	157
TM1SubsetElementDisplayEll	158
TM1SubsetElementDisplayLevel	159
TM1SubsetElementDisplayLine	159
TM1SubsetElementDisplayMinus	160
TM1SubsetElementDisplayPlus	161
TM1SubsetElementDisplaySelection	161
TM1SubsetElementDisplayTee	162
TM1SubsetElementDisplayWeight	162
TM1SubsetElementListByIndexGet	163
TM1SubsetElementListByIndexGetEx	163
TM1SubsetElementListByNamesGet	164
TM1SubsetInsertElement	165
TM1SubsetInsertSubset	165
TM1SubsetListGet	166
TM1SubsetListByNamesGet	167
TM1SubsetSelectByAttribute	168
TM1SubsetSelectByIndex	169
TM1SubsetSelectByLevel	170
TM1SubsetSelectByPattern	170
TM1SubsetSelectionDelete	171
TM1SubsetSelectionInsertChildren	172
TM1SubsetSelectionInsertParents	173
TM1SubsetSelectionKeep	173
TM1SubsetSelectNone	174
TM1SubsetSort	174
TM1SubsetSortByHierarchy.	175
TM1SubsetSubtract	176
TM1SubsetUpdate	176
TM1SystemAdminHostGet	177
TM1SystemAdminHostSet	177

TM1SystemBuildNumber	178
TM1SystemClose	178
TM1SystemGetServerConfig	179
TM1SystemOpen	180
TM1SystemProgressHookSet	180
TM1SystemServerClientName	181
TM1SystemServerConnect	182
TM1SystemServerConnectIntegratedLogin	183
TM1SystemServerDisconnect	183
TM1SystemServerHandle	184
TM1SystemServerName	185
TM1SystemServerNof	185
TM1SystemServerReload	186
TM1SystemServerStart	186
TM1SystemServerStartEx	187
TM1SystemServerStop	188
TM1SystemVersionGet	189
TM1UserKill	189
TM1ValArray	189
TM1ValArrayGet	190
TM1ValArrayMaxSize	191
TM1ValArraySet	191
TM1ValArraySetSize	192
TM1ValBool	192
TM1ValBoolGet	193
TM1ValBoolSet	193
TM1ValErrorCode	194
TM1ValErrorString	194
TM1ValIndex	195
TM1ValIndexGet	195
TM1ValIndexSet	196
TM1ValsUndefined	196
TM1ValsChanged	196
TM1ValsUpdatable	197
TM1ValObject	198
TM1ValObjectCanRead	198
TM1ValObjectCanWrite	199
TM1ValObjectGet	199
TM1ValObjectSet	200
TM1ValObjectType	200
TM1ValPoolCount	201
TM1ValPoolCreate	201
TM1ValPoolDestroy	201
TM1ValPoolGet	202
TM1ValPoolMemory	203
TM1ValReal	203
TM1ValRealGet	204
TM1ValRealSet	204
TM1ValString	204
TM1ValStringEncrypt	205
TM1ValStringGet	206
TM1ValStringMaxSize	206
TM1ValStringWMaxSize	207
TM1ValStringSet	207
TM1ValStringSetW	208
TM1ValStringSetUTF8	208
TM1ValStringUTF8MaxSize	209
TM1ValType	210
TM1ValTypeEx	211
TM1ValTypeIsString	211
TM1ValTypeIsBinary	211

TM1ViewArrayColumnsNof	211
TM1ViewArrayConstruct	212
TM1ViewArrayDestroy	213
TM1ViewArrayRowsNof	213
TM1ViewArrayValueByRangeGet.	214
TM1ViewArrayValueGet.	214
TM1ViewArrayValuePickListGet	215
TM1ViewArrayValuePickListByRangeGet	216
TM1ViewArrayValuePickListExists	217
TM1ViewCellValueGet	217
TM1ViewCellsValueGet	218
TM1ViewCreate	218
TM1ViewExtractCreate	219
TM1ViewExtractDestroy	220
TM1ViewExtractGetNext	220
TM1ViewListByNamesGet	221
TM1ViewListGet	222
Data Spreading Syntax	222

Chapter 5. IBM Cognos TM1 functions for Microsoft Visual Basic 225

Function types and naming conventions	225
TM1APIInitialize	225
TM1APIFinalize	226
TM1BlobClose	227
TM1BlobCreate.	227
TM1BlobGet.	228
TM1BlobOpen	229
TM1BlobPut.	229
TM1ChoreExecute	230
TM1ClientAdd	230
TM1ClientGroupAssign	231
TM1ClientGroupIsAssigned	232
TM1ClientGroupRemove	233
TM1ClientHasHolds	233
TM1ClientPasswordAssign	234
TM1ConnectionCheck	235
TM1ConnectionCreate	235
TM1ConnectionDelete	236
TM1ConnectionSynchronize	237
TM1CubeCellDrillListGet	237
TM1CubeCellDrillObjectBuild	238
TM1CubeCellSpread	239
TM1CubeCellSpreadStatusGet.	241
TM1CubeCellSpreadViewArray	244
TM1CubeCellValueGet	245
TM1CubeCellValueSet	245
TM1CubeCreate	246
TM1CubePerspectiveCreate.	247
TM1CubePerspectiveDestroy	248
TM1CubeShowsNulls	249
TM1DimensionCheck.	250
TM1DimensionCreateEmpty	250
TM1DimensionElementComponentAdd	251
TM1DimensionElementComponentDelete	252
TM1DimensionElementComponentWeightGet	253
TM1DimensionElementDelete	253
TM1DimensionElementInsert	254
TM1DimensionUpdate	256
TM1GroupAdd.	257
TM1ObjectAttributeDelete	257
TM1ObjectAttributeInsert	258

TM1ObjectAttributeValueGet	259
TM1ObjectAttributeValueSet	260
TM1ObjectAttributeValuesSet	261
TM1ObjectCopy	262
TM1ObjectDelete	263
TM1ObjectDestroy.	263
TM1ObjectDuplicate	264
TM1ObjectFileDelete	265
TM1ObjectFileLoad	265
TM1ObjectFileSave	266
TM1ObjectListCountGet	267
TM1ObjectListHandleByIndexGet.	268
TM1ObjectListHandleByNameGet	269
TM1ObjectPrivateDelete	270
TM1ObjectPrivateListCountGet	270
TM1ObjectPrivateListHandleByIndexGet	271
TM1ObjectPrivateListHandleByNameGet	272
TM1ObjectPrivatePublish	273
TM1ObjectPrivateRegister	273
TM1ObjectPropertyGet	274
TM1ObjectPropertySet	275
TM1ObjectRegister	276
TM1ObjectReplicate	277
TM1ObjectSecurityLock	278
TM1ObjectSecurityRelease	279
TM1ObjectSecurityReserve	279
TM1ObjectSecurityRightGet	280
TM1ObjectSecurityRightSet	281
TM1ObjectSecurityUnLock	282
TM1ProcessExecute	283
TM1ProcessExecuteEx	284
TM1ProcessExecuteSQLQuery	285
TM1ProcessVariableNamesValid	286
TM1RuleAttach.	286
TM1RuleCheck	287
TM1RuleCreateEmpty	288
TM1RuleDetach	288
TM1RuleLineGet	289
TM1RuleLineInsert	290
TM1ServerBatchUpdateFinish	291
TM1ServerBatchUpdateIsActive	292
TM1ServerBatchUpdateStart	292
TM1ServerLogClose	293
TM1ServerLogNext	294
TM1ServerLogOpen	295
TM1ServerOpenSQLQuery	296
TM1ServerPasswordChange	297
TM1ServerSecurityRefresh	298
TM1SQLTableGetNextRows	298
TM1SubsetAll	299
TM1SubsetCreateByExpression	300
TM1SubsetCreateEmpty	301
TM1SubsetElementDisplay	301
TM1SubsetElementDisplayEll	302
TM1SubsetElementDisplayLevel	303
TM1SubsetElementDisplayLine	304
TM1SubsetElementDisplayMinus.	304
TM1SubsetElementDisplayPlus	305
TM1SubsetElementDisplaySelection	306
TM1SubsetElementDisplayTee	306
TM1SubsetElementDisplayWeight	307

TM1SubsetInsertElement	307
TM1SubsetInsertSubset	308
TM1SubsetSelectByAttribute	310
TM1SubsetSelectByIndex	311
TM1SubsetSelectByLevel.	311
TM1SubsetSelectByPattern	312
TM1SubsetSelectionDelete	313
TM1SubsetSelectionInsertChildren	314
TM1SubsetSelectionInsertParents	315
TM1SubsetSelectionKeep	316
TM1SubsetSelectNone	316
TM1SubsetSort	317
TM1SubsetSortByHierarchy.	318
TM1SubsetSubtract	318
TM1SubsetUpdate.	319
TM1SystemAdminHostGet_VB	320
TM1SystemAdminHostSet	321
TM1SystemBuildNumber_VB	321
TM1SystemClose	322
TM1SystemOpen	322
TM1SystemServerClientName_VB	323
TM1SystemServerConnect	324
TM1SystemServerConnectIntegratedLogin.	325
TM1SystemServerDisconnect	326
TM1SystemServerHandle	326
TM1SystemServerName_VB	327
TM1SystemServerNof	328
TM1SystemServerReload	328
TM1SystemServerStart	329
TM1SystemServerStop	329
TM1SystemVersionGet	330
TM1ValArrayGet	330
TM1ValArrayGet	331
TM1ValArrayMaxSize	332
TM1ValArraySet	332
TM1ValArraySetSize	333
TM1ValBool	334
TM1ValBoolGet.	334
TM1ValBoolSet	335
TM1ValErrorCode	335
TM1ValErrorString_VB	336
TM1ValIndex	336
TM1ValIndexGet	337
TM1ValIndexSet	338
TM1ValIsUndefined	338
TM1ValIsUpdatable	339
TM1ValObject	339
TM1ValObjectCanRead	340
TM1ValObjectCanWrite	340
TM1ValObjectGet	341
TM1ValObjectSet	342
TM1ValObjectType	342
TM1ValPoolCount.	343
TM1ValPoolCreate.	343
TM1ValPoolDestroy	344
TM1ValPoolGet.	344
TM1ValPoolMemory	345
TM1ValReal	345
TM1ValRealGet.	346
TM1ValRealSet	347
TM1ValString	347

TM1ValStringEncrypt.	348
TM1ValStringGet_VB.	349
TM1ValStringMaxSize	349
TM1ValStringSet	350
TM1ValType.	350
TM1ViewArrayColumnsNof	351
TM1ViewArrayConstruct	352
TM1ViewArrayDestroy	353
TM1ViewArrayRowsNof	353
TM1ViewArrayValueGet.	354
TM1ViewCreate	355
TM1ViewExtractCreate	356
TM1ViewExtractDestroy.	357
TM1ViewExtractGetNext	357

Chapter 6. IBM Cognos TM1 properties 359

TM1AttributeType property	359
TM1BlobSize property	359
TM1ClientPassword property	360
TM1ClientStatus property	360
TM1ConnectionChoresUsing property	360
TM1ConnectionLastSyncTime property	361
TM1ConnectionLastSyncTimeStar property	361
TM1ConnectionUsername property	361
TM1ConnectionSyncErrorCount property	362
TM1ConnectionSyncPlanetToStar property.	362
TM1ConnectionSyncStarToPlanet property.	362
TM1CubeCellValueUndefined property.	363
TM1CubeDimensions property	363
TM1CubeLogChanges property	363
TM1CubeMeasuresDimension property.	363
TM1CubePerspectivesMaxMemory property	364
TM1CubePerspectivesMinTime property	364
TM1CubeReplicationSyncRule property	364
TM1CubeReplicationSyncViews property	365
TM1CubeRule property	365
TM1CubeTimeDimension property	365
TM1CubeViews property	365
TM1DimensionCubesUsing property	366
TM1DimensionElements property	366
TM1DimensionNofLevels property	366
TM1DimensionReplicationSyncSubsets property.	367
TM1DimensionSubsets property	367
TM1DimensionTopElement property	368
TM1DimensionWidth property	368
TM1ElementComponents property	368
TM1ElementIndex property	369
TM1ElementLevel property.	369
TM1ElementParents property	369
TM1ElementType property	370
TM1ObjectAttributes property.	370
TM1ObjectChangedSinceLoaded property	370
TM1ObjectLastTimeUpdated property	371
TM1ObjectMemoryUsed property	371
TM1ObjectName property	371
TM1ObjectNull property	372
TM1ObjectParent property	372
TM1ObjectRegistration property	372
TM1ObjectReplicationConnection property	373
TM1ObjectReplicationSourceObjectName property	373
TM1ObjectReplicationStatus property	373

TM1ObjectSecurityOwner property	374
TM1ObjectSecurityStatus property	374
TM1ObjectType property	374
TM1RuleErrorLine property	375
TM1RuleErrorString property	375
TM1RuleNoOfLines property.	376
TM1ServerBlobs property	376
TM1ServerBuildNumber property	376
TM1ServerChores property	377
TM1ServerClients property	377
TM1ServerConnections property	377
TM1ServerCubes property	377
TM1ServerDimensions property	378
TM1ServerDirectories property	378
TM1ServerGroups property	378
TM1ServerLogDirectory property.	378
TM1ServerNetworkAddress property	379
TM1ServerProcesses property	379
TM1SQLTableColumnNames property	379
TM1SQLTableColumnTypes property	379
TM1SQLTableNumberOfColumns property	380
TM1SQLTableNumberOfRows property	380
TM1SQLTableRowsetSize property	380
TM1SubsetAlias property	381
TM1SubsetElements property	381
TM1SubsetExpression property	381
TM1ViewColumnSubsets property	382
TM1ViewExtractComparison property	382
TM1ViewExtractRealLimitA property	382
TM1ViewExtractRealLimitB property	383
TM1ViewExtractSkipConsolidatedValues property	383
TM1ViewExtractSkipRuleValues property	383
TM1ViewExtractSkipZeroes property	383
TM1ViewExtractStringLimitA property	384
TM1ViewExtractStringLimitB property	384
TM1ViewFormat property	384
TM1ViewFormatString property	385
TM1ViewPreConstruct property	386
TM1ViewRowSubsets property	387
TM1ViewShowAutomatically property	387
TM1ViewSuppressZeroes property	387
TM1ViewTitleElements property	388
TM1ViewTitleSubsets property	388

Chapter 7. Performing data spreading with the IBM Cognos TM1 API 389

Spreading overview	389
Spreading internals	390
Spreading to a single leaf cell	390
Spreading to a consolidated cell	391
Spreading to a range	391
After spreading is complete	392
The spreading function arguments	392
Proportional spread, equal spread and repeat.	392
Clear	394
Percent change	396
Straight line	398
Growth%.	400
Relative proportional spread	402
Relative percent adjustment	404
Repeat leaves	406
Equal spread leaves	407

Applying holds.	409
Spreading control codes	410
Notices	415
Index	419

Introduction

This document is intended for use with IBM® Cognos® TM1®.

This manual describes the functions and features of the TM1 Application Programming Interface. The API is intended to give complete access to all the features and functionality of the TM1 OLAP engine. The API is to be used with C, C++, and Microsoft Visual Basic.

The API has been optimized for use in a networked environment. As a result, the conventions used vary markedly from those used in previous versions of the TM1 API.

IBM Cognos TM1 integrates business planning, performance measurement and operational data to enable companies to optimize business effectiveness and customer interaction regardless of geography or structure. TM1 provides immediate visibility into data, accountability within a collaborative process and a consistent view of information, allowing managers to quickly stabilize operational fluctuations and take advantage of new opportunities.

Finding information

To find IBM Cognos product documentation on the web, including all translated documentation, access one of the IBM Cognos Information Centers (<http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp>). Release Notes are published directly to Information Centers, and include links to the latest technotes and APARs.

Samples disclaimer

The Great Outdoors Company, GO Sales, any variation of the Great Outdoors name, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Accessibility features

This product does not currently support accessibility features that help users with a physical disability, such as restricted mobility or limited vision, to use this product.

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment,

promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Chapter 1. The IBM Cognos TM1 programming environment

This manual describes all the functions and features of the IBM Cognos TM1 API. It assumes you are familiar with the basic concepts of TM1.

The API is intended to give access to many of the features and functionality of the TM1 OLAP engine. The API is designed for use with Microsoft C, C++ and Microsoft Visual Basic. All the C functions are supported by Microsoft Visual Basic. However, some of the string and array value handling functions have a special version for Microsoft Visual Basic. You should use these functions if you are programming in Microsoft Visual Basic.

For C and C++ applications, include the header file TM1API.H. For Microsoft Visual Basic applications, include the file TM1API.BAS.

The API has been optimized for use in a networked environment. As a result, the conventions used vary markedly from those used in other Application Programming Interfaces you may be familiar with. For example, all values are passed as pointers or handles to *value capsules* that contain the actual data.

Location of the IBM Cognos TM1 API DLLs

There are two dynamic link libraries that must be loaded before you can use the IBM Cognos TM1 API:

- TM1LIB.dll
- TM1API.dll

These libraries must be loaded in the order that they are shown here. It is the programmer's responsibility to assure they are loaded in the correct order. There are three ways to do this:

- If the TM1API application resides in the same directory as the DLLs, the libraries will be correctly loaded at run time.
- If the PATH environment variable includes the <TM1_home>/bin directory, the libraries will be correctly loaded.
- You can explicitly load these libraries from within your application. For example, in Microsoft Visual Basic you can call the Windows system function LoadLibrary() to each of the DLLs. The LoadLibrary() function will have to be called twice: once for each library.

Setting a path to the DLLs

The following procedure assumes that you accepted the default settings during the installation, and that the IBM Cognos TM1 API .dll files are in C:\Cognos\TM1\bin.

Follow these steps to tell the compiler where to find the TM1 API .dll files.

Procedure

1. From the Task Bar choose **Start, Settings, Control Panel**.
2. Double-click the **System** icon.
3. Click on the **Advanced** tab in the System Properties window.

4. Click Environment Variables.
5. Click **PATH** in the User Variables list.
6. In the Value field, add the full name of the directory in which the TM1 API .dll files are located. The delimiter between paths is the semi-colon (;).
7. Click **OK**.

Note: The above procedure is accurate for Windows NT 4.0. Procedures for other Windows operating systems may vary slightly.

Supported compilers

The TM1 API supports the following compilers:

- Microsoft Visual C++ Release 4.0 and higher
- Microsoft Visual Basic Release 4.0 and higher

Servers

IBM Cognos TM1 servers control access to system and user databases. In TM1, a server is a *logical* object having no parent of its own. The server has children that include, but are not limited to, cubes and dimensions, blobs (Binary Large Objects), clients (users), connections, and processes.

Servers use control cubes to keep track of databases, clients (users), groups, connections and other information. For detailed information about server and other TM1 objects, see Chapter 3, “IBM Cognos TM1 objects,” on page 27.

A TM1 network is comprised of multiple local and remote servers.

Local servers

A local server runs on a users workstation. The local servers Admin Host machine is the user's workstation. If no Admin Server is running on the local machine, the local server starts one on the user's workstation. To start a local server, execute TM1S.EXE without configuration information or call TM1SystemServerStart.

Other users cannot access the local server by connecting to the admin host on that workstation.

Remote servers

A remote server controls access to public data objects. It can run on any workstation, including a users own workstation.

For a user to access the public data objects on a server, the user must be logged in to that server and must have at least READ rights. There are as many potential connections to a public server as there are server ports.

Procedure

1. Set up a configuration file specifying information for a specific server.
2. Create a shortcut to TM1S.EXE and add a -Z switch specifying the location of the server configuration file, TM1S.CFG. A server can be started by anyone.

The admin host and admin server

The IBM Cognos TM1 Admin Server is installed when you install TM1. This server acts as a rendezvous point for TM1 servers and TM1 clients, such as TM1 Architect and TM1 Perspectives. You must have an Admin Server running somewhere in your network in order for TM1 users to log in to the TM1 server. The TM1 Admin Server executable is called `tm1admsrv.exe`. The computer on which the Admin Server is running is called the TM1 admin host.

A TM1 API application gets information about available TM1 servers by calling the function `TM1SystemAdminHostSet`. As each new TM1 server starts, it registers itself with the Admin Server running on the admin host machine. By calling `TM1SystemAdminHostSet`, the API can ask the Admin Server for the following information:

- Which servers are available
- The protocol they use
- Their network address
- Port number servers are running on
- Their default packet size

For example, if Server A is the admin host that is running the Admin Server, you can do one of the following:

- Run a Client or API application on Server A and specify Server A as Admin Host by machine name or IP address.
- Run a Client or API application on Server B and specify Server A as the Admin Host by machine name or IP address.

If you do not specify an Admin Server and the server you are starting is local, TM1 assumes the users workstation is the Admin Host and automatically starts an Admin Server on it.

The role of an IBM Cognos TM1 server

The following table shows the major functions of the IBM Cognos TM1 server.

Time Frame	Server Functions
Start Time	Loads the names of all available permanent objects (cubes, dimensions, and so on) into memory. Registers with the Admin Server.
Normal Operation	Responds to client requests, loading objects, and performing calculations, consolidations and updates as required. Manages security, granting or denying access to server objects. Maintains a log of changes to the servers data.

Time Frame	Server Functions
Upon Termination	Save all data if instructed. Unregisters with the Admin Server.

Multitasking and symmetric multiprocessor support

The IBM Cognos TM1 server manages thread contention using reader/writer locks at the database level. Reader / writer locks allow multiple readers to access the database at the same time. However, only one write request can be processed at a time.

In general, functions that update data and metadata require exclusive access to the database. These functions lock the database when they begin processing, and unlock the database when they have completed processing.

Functions that read data from the database are serviced simultaneously.

In a multiprocessor environment, the TM1 server can take advantage of the operating systems ability to balance simultaneous calculations across processors, resulting in an increase in server response time.

IBM Cognos TM1 server performance

There are situations that may cause performance problems with IBM Cognos TM1. One of the most common is to have a write request arrive at the server while the server is processing a series of very long calculations. The write request locks the database while the calculations complete, preventing any ensuing requests from processing.

For example, if the TM1 server is processing a long calculation when the write request arrives, the write request will not be processed until the long calculation has completed. All ensuing requests must wait for both the long calculation and the write request to complete. This can temporarily slow performance.

Write requests can also slow down long calculations because they may invalidate previously calculated results on which the calculations rely. This forces some calculations to be repeated.

TM1 performance can sometimes be improved by modifying parameters in TM1S.cfg. Contact your IBM Cognos support representative for more information.

Sample code for the IBM Cognos TM1 API

We provide sample code for both Microsoft Visual Basic and C. The samples are located in the IBM Cognos TM1 API installation directory.

Chapter 2. IBM Cognos TM1 programming concepts

There are a number of issues that every IBM Cognos TM1 programmer encounters when using the TM1 API.

This section describes some of those issues. The following topics are discussed:

- System Functions - Every TM1 program must use the TM1 system functions to communicate with the TM1 server, and manage its connection during program execution. This section describes those functions.
- Simple TM1 Data Types - Description of the types of values supported by the TM1 API.
- Values - Most TM1 functions use values to store and manage information. This section describes the types of values used by the TM1 API, and how those values are manipulated.
- Security - The security of your data is of paramount importance. This section introduces two TM1 objects dedicated solely to security: Clients and Groups.
- Backup and Recovery - The TM1 server offers functions that allow full recovery of all cubes in the event the TM1 server is not properly brought down. This section describes the functions that allow you to recover your data in the event of a crash.

This section also includes:

- Creating a TM1 Project in C or C++
- Creating a TM1 Project in Microsoft Visual Basic

System functions

The system functions control the connections between the IBM Cognos TM1 client application and the API, and between the TM1 API and the servers.

This section describes those functions.

Connecting to the API

When you use the IBM Cognos TM1 API in either C or in Microsoft Visual Basic a, then you must use a specific call sequence.

The following sections describe the function call sequence for:

- Connecting to the TM1 API in C
- Connecting to the TM1 API in Microsoft Visual Basic

Connecting to the IBM Cognos TM1 API in C

Every IBM Cognos TM1 API application written in C must call the following functions, in a specific order, to initialize a server session.

This is the function order:

- TM1APIInitialize - This function initializes memory for use by the API. It is important for all TM1 API applications, but is particularly important for applications that must support many users. We recommend starting every TM1 API application with this function.

- **TM1SystemOpen** - Returns a user handle (TM1U), which is used throughout your application to identify which TM1 client is making requests to the server. The user handle is used to build TM1 value pools (TM1P).
- **TM1SystemAdminHostSet** - The TM1 admin server serves as a rendezvous point for clients and servers in a TM1 system. In order to log in to a particular TM1 server through the API, you must call **TM1SystemAdminHostSet**, passing the machine name of a TM1 admin host. Once you have set the admin host for the session, you can log in to any TM1 server that has registered with that admin host.
- **TM1ValPoolCreate** - Value pools are used to pass data between the TM1 server and the TM1 clients. Most TM1 API functions require a value pool.
- **TM1SystemServerConnect** - This function logs your application into TM1. This function works for all authentication methods except integrated login.

Note: If your TM1 server is configured to accept Integrated Login (the **IntegratedSecurityModeparameter** in **tmls.cfg** is set to either 2 or 3.) you can connect to the server using the TM1 API function, **Tm1SystemServerConnectIntegratedLogin**.

The following code sample shows a simple program that initializes the API and logs in.

```
#include <stdio.h>
#include "Tmlapi.h"
#include "string.h"
int main(int argc, char* argv[])
{
    TM1V hServer = NULL;
    TM1U hUser;
    TM1V vServerHandle=NULL;
    TM1V vResult;
    TM1P hPool1, hPool2;
    //
    // All TM1 Applications start with TM1APIInitialize()and
    // TM1SystemOpen().
    //
    TM1APIInitialize();// Initializes memory for the API
    hUser = TM1SystemOpen();// Retrieves a TM1 API session handle
    //
    // Call TM1SystemAdminHostSet to establish the AdminServer to which
    // you want to connect.
    //
    char * sAdminHost = "rjordon";
    TM1SystemAdminHostSet (hUser,(CHAR *) sAdminHost);
    //
    // Create Value Pools from strings so that we can log in.
    //
    hPool1 = TM1ValPoolCreate(hUser);
    hPool2 = TM1ValPoolCreate(hUser);
    char * sUserName = "admin";
    char * sServerName = "sdata";
    char * sPassword = "apple";
    TM1V vUserName = TM1ValString(hPool1, (CHAR *) sUserName, 10);
    TM1V vServerName = TM1ValString(hPool1, (CHAR *) sServerName, 10);
    TM1V vPassword = TM1ValString(hPool1, (CHAR *) sPassword, 10);
    //
    // TM1SystemServerConnect logs in to TM1. This function returns
    // a TM1 server object
    //
    TM1V vServerObject = TM1SystemServerConnect(hPool1, vServerName,
    vUserName, vPassword);
    TM1V vsServerName = TM1ObjectPropertyGet (hPool2, vServerObject, TM1ObjectName());
};
```

```

sServerName = (char *) TM1ValStringGet(hUser, vsServerName);
// Logging out and cleaning up.
// Best practice dictates that all pool handles used in your application
// should be destroyed with TM1ValPoolDestroy.
vResult = TM1SystemServerDisconnect(hPool2, vServerName);
// Destroy value pools
TM1ValPoolDestroy(hPool1);
TM1ValPoolDestroy(hPool2);
TM1SystemClose(hUser);
TM1APIFinalize();
return 0;
}

```

Connecting to the IBM Cognos TM1 API in Microsoft Visual Basic

Every IBM Cognos TM1 API application written in Microsoft Visual Basic must call the following functions, in a specific order, to initialize a server session.

This is the function order:

```

Private Sub Form_Load()
Dim sServerName As String
Dim sUserName As String
Dim sPassword As String
Dim hUserHandle As Long
Dim pPoolHandle As Long
Dim vPassword, vServerName, vUserName As Long
Dim vStringLength As Long
Dim RetVal As String * 75

' Each Microsoft Visual Basic application begins by calling TM1APIInitialize,
' TM1SystemAdminServerHostSet, and TM1SystemServerConnect.

TM1APIInitialize
hUser = TM1SystemOpen()
TM1SystemAdminHostSet hUser, "rjordon"
pPoolHandle = TM1ValPoolCreate(hUser)

' We have to take the strings containing the login information (such
' as the user name and password) and turn them into TM1 value
' capsules. First establish the maximum length of the string as
10
' characters.

vStringLength = TM1ValIndex(pPoolHandle, 10)

Next, use this string length to build value capsules for the
' user name, password, and TM1Server name. We can reuse the pool
' handle for these functions.

vUserName = TM1ValString(pPoolHandle, "admin", vStringLength)
vPassword = TM1ValString(pPoolHandle, "apple", vStringLength)
vServerName = TM1ValString(pPoolHandle, "sdata", vStringLength)
vServerHandle = TM1SystemServerConnect(pPoolHandle, _
vServerName, vUserName, vPassword)

' To log out and disconnect from the API, you must
' call TM1SystemServerDisconnect, TM1SystemClose, then TM1APIFinalize.

' In addition, best practice dictates that all TM1 Value Pools used
' in your program be destroyed by calling TM1ValPoolDestroy().

vResult = TM1SystemServerDisconnect(pPoolHandle, vServerName)
.....

```

```
' Add Code to delete all TM1 Value Pools here.
.....
TM1ValPoolDestroy (pPoolHandle)
TM1SystemClose hUser
TM1APIFinalize
End Sub
```

Disconnecting from the API

You can disconnect from the IBM Cognos TM1 API.

To disconnect from the API, use the following functions in this order:

- TM1SystemServerDisconnect- Logs out of TM1 and ends the client session.
- TM1ValPoolDestroy - Call this function once for each pool handle you created with TM1ValPoolCreate during the API session.
- TM1SystemClose - Ends the API session.
- TM1APIFinalize - Cleans up memory structures used by the API. Using this function is highly recommended if your application is being run by many simultaneous users.

It is considered best practice to destroy all the TM1 pool handles you create by calling TM1ValPoolDestroy for each handle.

Managing server connections

You can determine what servers are available with the IBM Cognos TM1 API.

To determine what servers are available, the API provides the following functions:

- TM1SystemServerReload loads information from the TM1 admin server into the TM1API. You should call this function before you call TM1SystemServerNof.
- TM1SystemServerNof loads the list of available servers (servers to which the API is connected or which are available for connection) and returns the number of items on the list.
- TM1SystemServerName returns the name of an available server, given an index within the list.
- TM1SystemServerHandle accepts a server name as an argument, and returns the handle to the server, or zero if the user is not connected to the server.
- TM1SystemServerConnect connects the client program to a server and returns the handle to it.
- TM1SystemServerDisconnect disconnects the client program from the server.

Setting the admin host server

The API gets information about available IBM Cognos TM1 servers from a special admin host server called Tm1admsrv.exe.

Before you call any of the TM1SystemServer functions, you must set the name of the Admin Host server by calling the function TM1SystemAdminHostSet.

For more information, see “The admin host and admin server” on page 3.

Progress callback functions

Progress callback functions give you feedback about the progress of a task.

Normally, IBM Cognos TM1 API functions return immediately with a value. There are functions, however, that may take a long time to complete and which provide a periodic report on their progress. The feedback from such functions can be processed and displayed to the user using a *progress callback* function.

To setup a progress callback, use the function `TM1SystemProgressHookSet`.

To clear a previous progress callback, call the function `TM1SystemProgressHookSet` with a NULL pointer argument. See Chapter 4, “IBM Cognos TM1 functions for C,” on page 53 for more information on using this function in C.

Requesting the version of the system

The function `TM1SystemVersionGet` returns an integer indicating the current version and release of the IBM Cognos TM1 API.

Please refer to the IBM Cognos TM1 *Release Notes* for the version number of the software you are using.

Thread safety

Multiple threads can open multiple connections to the IBM Cognos TM1 API. Connections can be shared between threads.

However, once a thread calls an API function with a given user handle, all other threads that call the API with the same user handle will be suspended until the first one returns.

Values

IBM Cognos TM1 programs use value pools to move data between client applications and servers.

Value pools contain a set of one or more *value capsules*, which are a network-optimized data type specific to TM1. Value capsules contain simple data types, such as integers, arrays or strings.

Handles

The client application uses handles to manipulate IBM Cognos TM1 data.

In the TM1 API, there are many types of handles, all of which have specific data types defined in `TM1API.H`. In Microsoft Visual Basic, there are no predefined data types for handles. All handles are declared as long integers.

The following table shows some of these handles, and how they are declared in C and Microsoft Visual Basic applications.

Handle Definition	Description	Declaration
TM1U	TM1 User Handle	TM1U hUser; //c declaration dim hUser as long //VB declaration

Handle Definition	Description	Declaration
TM1P	TM1 Pool Handle	TM1P hPool; //c declaration dim hPool as long //VB declaration
TM1V	TM1 Value Capsule	TM1V vDimension; //c declaration dim vDimension as long //VB declaration

TM1 client applications retrieve data from the TM1 server and by calling functions using pool handles (TM1P). Once the client application receives the value pool, it calls functions to retrieve the value capsules (TM1V) from the value pool. Once the TM1 value capsules have been retrieved, the application can extract the simple TM1 data types from the value capsules.

Simple IBM Cognos TM1 data types

The IBM Cognos TM1 API supports the following types of values:

Value	Description
TM1_REAL	A double precision IEEE floating point number.
TM1_STRING	A character string whose length does not exceed 64K bytes.
TM1_INDEX	A 32 bit unsigned integer.
TM1_BOOL	A logical value stored as 1 to indicate TRUE, 0 to indicate false.
TM1_OBJECT	A structure used to refer to objects managed by the API. Typically an object contains information about the server in which the object resides, and an index to locate the object within the server.
TM1_ERROR	A structure containing typically an error code and an error message string.
TM1_ARRAY	An array of values.

Value handling functions

The IBM Cognos TM1 API provides functions to construct value pools and value capsules, find out the type of data they contain, and to extract their contents.

These functions are described in the following sections.

Value capsule construction functions

Value capsule construction functions take raw values such as strings or floating point numbers, construct a value capsule in a value pool, and return a handle to it.

The construction of value capsules is mostly used in situations in which data is to be sent from the user to the server. The functions are:

```
TM1ValReal  
TM1ValString  
TM1ValBoolean  
TM1ValObject  
TM1ValIndex  
TM1ValArray
```

Pool creation function

Before you can construct value capsules or retrieve data from a server, you must create a value pool.

To create a value pool, use the function `TM1ValPoolCreate`. This function takes a user handle as an argument and returns a pool handle, which is then used to construct value capsules and to receive values from a server.

Value typing

The function `TM1ValType` returns an integer indicating the type of the data contained in a value capsule.

These functions are mostly used to determine the type of a value capsule being sent from the server to the user. The type can be one of the following:

```
TM1V_TYPE_REAL  
TM1V_TYPE_STRING  
TM1V_TYPE_INDEX  
TM1V_TYPE_BOOL  
TM1V_TYPE_OBJECT  
TM1V_TYPE_ERROR  
TM1V_TYPE_ARRAY
```

Simple value extraction

The simple value extraction functions take the user handle, and a value handle, and return the corresponding raw value.

These functions are mostly used to extract data from value capsules that are returned from the server to the user. They are:

```
TM1ValRealGet  
TM1ValBoolGet  
TM1ValIndexGet
```

Error code extraction

The function `TM1ValErrorCode` returns a 32-bit code associated with the error.

Error string extraction

The function `TM1ValErrorString` returns a string associated with a given error code.

String handling

The string handling functions extract information from a string value capsule.

They are:

- `TM1ValStringMaxSize` returns the largest string size that this value capsule can hold. The actual size of the data in the capsule is less than or equal to the maximum length.
- `TM1ValStringGet` returns the string contained in a TM1 value capsule.

Object handling

Objects may be stored in a structure of type `TM1OBJECT` for future use.

Objects are only valid as long as the connection to the server on which they reside is active. The object handling functions extract information from an object value capsule. They are:

- `TM1ValObject` constructs an IBM Cognos TM1 value capsule from an object handle.
- `TM1ValObjectGet` returns the object from the value capsule.
- `TM1ValObjectSet` establishes a new object in a value capsule.
- `TM1ValObjectCanRead` determines if you have read rights to the object.
- `TM1ValObjectCanWrite` determines if you have write rights to the object.
- `TM1ValObjectType` returns the type of object. Global values for object types are listed in `Tm1api.h`.

The first three functions (`TM1ValObject`, `TM1ValObjectSet`, and `TM1ValObjectGet`) are used with data structures of type `TM1OBJECT`. The other functions deal with value capsules of the object. We strongly encourage using value capsules when working with objects, as it provides more efficient server communication.

Array handling

Array values are arrays of value handles.

The following functions are used to work with array values:

- `TM1ValArray` creates an IBM Cognos TM1 value capsule from an array
- `TM1ValArraySize` returns the size of the array.
- `TM1ValArrayGet` returns one value from the array.
- `TM1ValArrayMaxSize` returns the maximum number of components of an array value capsule. The actual number of array elements in the value capsule is returned by the function `TM1ValArraySize`.

Updating value capsules

The value update functions change the contents of previously created value capsules.

They are:

- `TM1ValBoolSet`
- `TM1ValIndexSet`
- `TM1ValObjectSet`
- `TM1ValRealSet`
- `TM1ValStringSet`
- `TM1ValArraySet`

The maximum length of array and string values is established when they are constructed. This length must not be exceeded when they are updated.

Pending values

Because the IBM Cognos TM1 API is designed to operate on a network, it provides facilities for maximizing packet size and optimizing performance.

With the API you can issue a number of requests before any actual values are returned. Normally, functions return a *pending value*. Such pending values are not assigned an actual value until the next packet is sent to the server and the answer received.

You can use pending values to get other values without waiting for them to be resolved.

For example, suppose you have the handle to a cube, and you want to know how many dimensions it has. This requires two calls, `TM1CubeDimensions()`, and `TM1ObjectListCountGet()`. The result of the first function is used as an argument to the second. The following code sample retrieves the first cube handle from the server, and counts the number of dimensions in the cube.

```
vCubeHandle = TM1ObjectListHandleByIndexGet(hPool1, vServerObject, TM1ServerCubes(),
    TM1ValIndex(hPool3, 1));
vCount = TM1ObjectListCountGet(hPool2, vCubeHandle, TM1CubeDimensions()
);
iCount = TM1ValIndexGet(hUser, vCount);
```

The API allows you to pass `TM1CubeDimensions` as an argument (and pending value) to the function `TM1ObjectListCountGet`. The server has no problem handling the request because, unlike the client, it already knows the result of the first function before it evaluates the second one.

All pending values are resolved any time you request the type or the contents of a value capsule. When this happens, the API sends any partial request packets to their respective servers and stores the answers received in the appropriate value pool.

Managing value pools

As values are constructed and returned to the user, value pools consume memory. Periodically, the user should release this space.

The following functions are provided for this purpose:

- `TM1ValPoolMemory` returns the memory consumed by a value pool.
- `TM1ValPoolDestroy` releases all memory used by a value pool. Note that all handles referring to values in the value pool become invalid.
- `TM1ValPoolCreate` creates a new value pool.

Value pools can be used for optimizing network performance by retrieving a large sequence of values. This approach minimizes the number of packets that are transmitted between the client and the server.

The API facilitates managing such value sets by means of the function `TM1ValPoolGet`. This function retrieves the values that have been requested by index. The following steps show a typical way of retrieving large sets of values.

Procedure

1. Create a value pool using `TM1ValPoolCreate`.

2. Issue a series of requests. As each request is executed, the result is assigned an index within the pool.
3. Retrieve the results, sequentially or randomly using the `TM1ValPoolGet` function.
4. Finally destroy the pool using the `TM1PoolDestroy` function to recover all its space.

Results

Use `TM1ValPoolCount` to find out the current number of values in a pool.

Object attributes

IBM Cognos TM1 supports the creation and maintenance of attributes for objects.

An attribute is like a property, but is defined by the user. Attributes associate a data value with each object in a list of objects. For example, you can define an attribute that is associated with all the cubes on a server, or all the elements of a dimension.

The following functions allow you to create and manage attributes:

- `TM1ObjectAttributeInsert` allows you to create an attribute. The attribute is assigned one of the data types supported by TM1.
- `TM1ObjectAttributeDelete` allows you to destroy an attribute.
- `TM1ObjectAttributeGet` retrieves the value of an attribute for a particular object in the list of objects.
- `TM1ObjectAttributeSet` updates the value of an attribute for a particular object in the list of objects.

Security

Most operations within the IBM Cognos TM1 API are governed by the TM1 security system.

Security is implemented using two security objects: *Clients* and *Groups*.

Security rights levels are assigned by object to each group. Clients are assigned to one or more groups. The security level of a client for an object is the highest level, for that object, of any group to which the client belongs. For example, suppose group Doc has WRITE permissions for the cube Books, for which group Support has READ permissions. If client Jane is a member of both groups, Jane has WRITE permissions for the cube Books, since WRITE is a higher permission level than Read.

See the IBM Cognos TM1 *Operations Guide* for more information on TM1 security.

Security levels

The IBM Cognos TM1 API supports several security levels.

For a description of security levels, refer to the IBM Cognos TM1 *Operations Guide*. The API supports the following security levels:

Value	Description
NONE	The client has no access to the object. For display and listing purposes it will be treated as if it does not exist.
READ	The client can look at the properties of the object.
WRITE	The client can update the updateable properties of the object.
RESERVE	The client can temporarily revoke the WRITE privileges of all other clients.
LOCK	The client can permanently suspend the WRITE privileges for all users.
ADMIN	The client has WRITE privileges at all times, and can unlock and delete objects.

Groups

A group is an object used to control access to IBM Cognos TM1 objects.

Rather than assign rights for an object directly to a client, the group object allows assigning the same set of rights to multiple clients.

Groups are implemented as minor objects and are created on a particular server. The following functions manipulate groups:

- TM1GroupAdd creates a new group.
- TM1ObjectDelete deletes an existing group.
- TM1 implicitly defines a special group named ADMIN, which has ADMIN privileges to all objects.

Clients

When you connect to an IBM Cognos TM1 server, you must supply a client id and a password.

In standard TM1 security, the TM1 server maintains a list of clients, along with security information that includes the client's password. If your TM1 server is configured for LDAP authentication or Integrated Login, passwords may be stored outside your TM1 server.

In some parts of the TM1 software, the term *user* is used. In TM1, a *client* is a named TM1 object used to establish a session with the TM1 server. A *user* is a human being who is using a TM1 client. The delineation is important, because multiple users can log in using the same client name. Also, a single user can log in to two different servers using two different client names.

Clients are implemented as minor objects and are created on a particular server. The following functions manipulate clients:

- TM1ClientAdd creates a new client.
- TM1ObjectDelete deletes an existing client.

Assigning clients to groups

You can use certain functions to assign clients to groups.

The following functions are available:

- `TM1ClientGroupIsAssigned` returns TRUE if a client is in a given group. Otherwise, `TM1ClientGroupIsAssigned` returns FALSE.
- `TM1ClientGroupAssign` assigns a client to a group.
- `TM1ClientGroupRemove` removes a client from a group.

Assigning rights to objects and groups

You can use certain functions to assign and retrieve a group's rights to an object.

The following functions are available:

- `TM1ObjectSecurityRightGet` returns the current rights level of a group for an object.
- `TM1ObjectSecurityRightSet` assigns a rights level to a group for an object.

Meta security

Security assignment is itself subject to security constraints.

In order to change the access privileges to a group and an object, you must have ADMIN privileges to both. If you are a member of the ADMIN or SecurityAdmin group, you are free to do any assignment of rights. However, the assignment of security rights can be delegated to any portion of the data in a server by appropriately granting ADMIN privileges to specific objects.

Managing locks and reservations

If you have rights, then you can use certain functions to lock or reserve objects.

The following functions are available:

- `TM1ObjectSecurityReserve` reserves an object. When an object is reserved, only the client who reserved it or a client with ADMIN rights may update the object.
- `TM1ObjectSecurityRelease` releases a reserved object.
- `TM1ObjectSecurityLock` locks an object.
- `TM1ObjectSecurityUnLock` unlocks an object. You must have ADMIN privileges to the object in order to perform this function.

A member of the ADMIN or DataAdmin group can unlock any object.

A user with LOCK privilege can unlock an object that he/she locked.

You can use the Boolean properties `TM1ObjectSecurityIsReserved` and `TM1ObjectSecurityIsLocked` to determine if an object is reserved or locked.

Determining access rights for a client

The property function `TM1ObjectSecurityClientRight` returns your access rights to an object.

You must have at least READ rights to the object in order to access this property. If you do not have READ rights, `TM1ObjectSecurityClientRight` returns an error.

Your rights to an object are the highest rights assigned to any of the groups to which you belong.

Whenever an object value is retrieved through the API, the value contains flags that tell you whether you have read or write access to the object.

The function `TM1ValObjectCanRead` returns a Boolean 1 if one of the groups to which you belong has READ (or higher) rights to the object. Otherwise, `TM1ValObjectCanRead` returns zero.

`TM1ValObjectCanWrite` returns a Boolean 1 if one of the groups to which the client belongs has WRITE (or higher) rights to the object, provided that the object is not reserved or locked. Otherwise, `TM1ValObjectCanWrite` returns zero.

Error handling

Use the function `TM1ValType` to determine if the value returned from a function is an error code.

Examining error values

Error values consist of an error code. To retrieve an error code, use the function `TM1ValErrorCode`.

Error codes

Error codes can be checked against constants that are exported by the API and are named according to a scheme.

The scheme looks like this:

`TM1Error...`

The following error codes can be returned by most API functions:

- `TM1ErrorSystemUserHandleIsInvalid`
- `TM1ErrorSystemValueIsInvalid`
- `TM1ErrorObjectHandleIsInvalid`

Error codes that are specific to one or a few functions are described with each individual function.

Error code functions are listed in `Tm1api.h`.

API error codes for data reservations

The following table describes the possible error codes that can be returned by the IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) C API functions for data reservations.

Error	Description
<code>TM1ErrorCubeNumberOfKeysInvalid</code>	Number of elements doesn't match the number of cube dimensions.
<code>TM1ErrorObjectHandleInvalid</code>	Cube, Client, or Element handle does not map to an existing object.
<code>TM1ErrorCubeKeyInvalid</code>	The element supplied doesn't match an element in the dimension at that position. The element supplied is a UDC.

Error	Description
TM1ErrorObjectSecurityNoReserveRights	Capability to use reservation is not granted.
TM1ErrorObjectSecurityNoAdminRights	Attempt to release a reservation when not the owner and without the override capability being granted.

Backup and recovery

The IBM Cognos TM1 server offers functions that allow full recovery of all cubes in the event the TM1 server is not properly brought down.

Cubes are maintained in RAM and are not saved to disk until either the TM1 server is brought down, or the TM1 server administrator chooses File, Save Data All from TM1 Architect.

The TM1 server's backup and recovery functions provide for restoration of the cubes even if a system crash occurs before the cubes are saved. The core of the backup and recovery system is the transaction log file.

Note: Transaction logs are not available on Local servers. They are available only on remote TM1 servers, which are servers that are available to clients throughout your network.

The transaction log file

The IBM Cognos TM1 server logs all transactions made by each user in the transaction log file.

For example, suppose USR2 makes a change in a number in the xbudget cube on the TM1 server. When this happens, a record is automatically added to the transaction log file. The record contains the following information:

- The date and time the change was made.
- The name of the user who made the change.
- The name of the cube in which the change was made.
- The value before it was changed.
- The value after the change.
- The intersection of the cube that was changed.

This file is used to recover crashed TM1. It also provides a complete audit trail of all changes to the server. ChangeSets are stored in the transaction log to implement the Undo/Redo feature.

Transaction Logging must be turned on using a configuration setting. When transaction logging is enabled, access the transaction log file, by right-clicking the TM1 server in TM1 Architect and choose **View Transaction Log**.

ChangeSets

A ChangeSet is a collection of cell changes actions stored in the transaction log that share a common identifier.

When requesting an UNDO, this identifier is provided to the roll back function and all changes with that identifier will be backed out. The logfile is read in reverse order to improve performance. Note that ChangeSets cannot nest. If you call ChangeSetBegin() without having called ChangeSetEnd() from a previous Begin(), you will get an error returned.

The End of Changeset marker contains a count marker that stores the number of items that were modified in that changeset. When the log file is read by the undo operation, it encounters this end change set marker first, parses off this count, then continues the search (in reverse order) for the quantity of changes indicated by this value. Searching will terminate when this count has been achieved, thereby alleviating any unnecessary log file scanning.

Treatment of time

All times in the log file are stored in GMT (Greenwich Mean Time).

Time stamps in records and file names are written as character strings of the form: YYYYMMDDhhmmss. For example January 2, 2001 at 2:30 PM GMT is written:

20010102143000

Naming conventions

When the IBM Cognos TM1 server is loaded, it automatically creates a file called TM1S.LOG.

When a client changes a number in a cube, the TM1 server opens this file, writes a new record, and closes the file again. When the TM1 server is brought down, this file is renamed using the current GMT time to TM1SYYYYMMDDHHmmss.LOG.

Log files can take up a substantial amount of disk space after the TM1 server has been running for some time. You will need to remove old log files from your disk every so often, depending on the volume of changes being made and the size of your disk. If you have a need for archival information, you may want to back up these files before erasing them.

Log file structure

The transaction log file is a standard comma-delimited ASCII file.

It contains the following fields:

- Date/Time, as a numeric string in GMT.
- Name of client who performed the transaction (for example, updating a value in a cube).
- Transaction type: Currently always a "C" to indicate a cube cell change.
- Old value (value before update).
- New value (value after update).
- Name of the cube updated.

Dimension elements that define the intersection updated.

Automatic recovery

When an IBM Cognos TM1 server ends operation abnormally, it leaves a TM1S.LOG file on disk.

When the server is re-started, it looks for the TM1S.LOG file. If it finds it, then it will apply all the changes to the cubes that it has in memory, subject to operator approval.

Accessing the log files

The API provides functions to access the log files that currently exist for the server.

It logically concatenates the current log file (TM1S.LOG) with any prior files that exist in the data directory, and presents the user with a chronological sequence of all transactions. The following functions allow you to access and manipulate log files:

- The function `TM1ServerLogOpen` allows you to access the log files as of a certain time in the past.
- `TM1ServerLogNext` retrieves the next item of data (field) from the log file.
- `TM1ServerLogClose` terminates access to the log file.

IBM Cognos TM1 also includes a message log file that contains error messages and system alerts. The following functions allow you to access and manipulate message log files:

- The function `TM1ServerMessageLogOpen` allows you to access the message log files as of a certain time in the past.
- `TM1ServerMessageLogNext` retrieves the next item of data (field) from the message log file.
- `TM1ServerMessageLogClose` terminates access to the message log file.

Logging changes to dimensions

Whenever a dimension is updated, a copy of the new dimension is written to disk as a file.

The file has the following name:

`TM1S}{dimname}YYYYMMDDHHmmss.DIM.`

In addition, a record with the following fields is written to the transaction file.

- Date/Time, as a numeric string in GMT - corresponding to the time stamp in the dimension log file.
- Name of client who performed the transaction (i.e., updated the dimension).
- Transaction type: D to indicate a changed dimension.
- The name of the dimension being changed.

Creating an IBM Cognos TM1 project in C or C++

This section describes how to create a very simple IBM Cognos TM1 program in Microsoft Developer's Studio.

To create a Microsoft Visual Studio project, you need to do the following:

- Generate a skeleton console project. The simplest one to create is a WIN32 console application. We will use the program that Microsoft Visual C++ supplies, Hello World, as an example.
- Make the support files visible to the compiler. This includes setting paths to the header files, libraries, and dynamically linked library (dll) files.

- Modify the program to include code that will test whether the compiler sees the paths. This includes entering code in the C++ file that points to the TM1 header file and code that uses TM1 library and .dll files.
- Compile and link the test program to verify that the compiler sees the include file, library, and .dll files.

Generating a console project

Follow these steps to build an IBM Cognos TM1 project with Microsoft Visual C++ 6.0.

Procedure

1. Open Microsoft Visual C++.
2. From the menu bar, choose **File, New**.
3. Click **Project**.
4. Choose **Win32 Console Application**.
5. Specify a project name in the Project field. For this example, use the name tm1test1.
6. Click **OK**.
7. In the Win32 Console Application - Step 1 of 1 window, choose **A "Hello World !"application**.
8. Click **Finish**. You will get a New Project Information window with a description of the skeleton project the Win32 Console Application will create for this project.
9. Click **OK**.
10. Your new project will appear in the workspace pane.

Setting paths

You can compile and link an IBM Cognos TM1 program.

To successfully compile and link a TM1 program, you must do the following:

- Within Developer's Studio, set a path to the location of the TM1 API header file (TM1API.H). Similarly, set a path to the location of the TM1 import library (TM1API.LIB).
- Add the name of the TM1 import library (TM1API.LIB) to the list of library modules for the project.
- Add the location of the TM1 API dynamic link library to your PATH environment variable. This allows the program to find the tm1api.dll when you run the program. You set this environment variable through the Windows NT System control panel.

Setting a path to the IBM Cognos TM1 API header and library files

When you install the IBM Cognos TM1 API, the TM1API.H file is copied to a certain directory.

The directory is: TM1S7\API directory. To make sure that the TM1 project can find this header file at compile time, follow these steps:

Procedure

1. From the Microsoft Visual C++ menus, choose **Tools, Options**.
2. Click the **Directories** tab.

3. Choose **Include files** from the Show Directories for: drop-down list.
4. Add the directory that contains your TM1 header file to the list in the Directories box. For example, add the C:\TM1S7\API path.
5. Choose **Library files** from the Show Directories for: drop-down list.
6. Add the directory that contains your TM1 library file to the list in the Directories box. For example, add the C:\TM1S7\API path.
7. Click **OK**.
8. Choose **Project, Settings** from the Microsoft Visual C++ menu.
9. Click the **Link** tab.
10. Enter `tm1api.lib` in the Object/library modules field followed by a space.
11. Click **OK**.
12. Choose **File, Save all** from the Microsoft Visual C++ menu.

You should now be able to compile and link `tm1test1.cpp` without error.

Adding test code

You can add test code to have the compiler see the new library and include files.

To test that the compiler sees the new library and include files, add the following code to your test program, `tm1test1.cpp` by following the steps below.

Procedure

1. Double click on the **tm1test1** files in the Workspace pane. Double click on **Source Files**. Click on **tm1test1.cpp** to open it in the Editor area.
2. Modify the code as shown here:

```
#include "stdafx.h"
#include <tm1api.h>
int main(int argc, char* argv[])
{
    TM1U hUser;
    TM1APIInitialize();
    hUser = TM1SystemOpen();
    printf("Hello World!\n");
    TM1SystemClose(hUser);
    TM1APIFinalize();
    return 0;
}
```

Building and running the program

To build the program, you have to compile and link.

Procedure

1. Click on **Build, Build tm1test1.exe** to compile and link **tm1test1.cpp**.
2. Click on **Build, Execute tm1test1.exe** to verify that it writes "Hello world!" in a DOS console window.
3. Click on **File, Save All** from the Microsoft Visual C++ menu to save all the files in the project.

Creating an IBM Cognos TM1 project in Microsoft Visual Basic

Follow these steps to build an IBM Cognos TM1 project with Microsoft Visual Basic.

Procedure

1. Open Microsoft Visual Basic. A blank workspace window appears.
2. Choose **File, New Project** from the menu bar.
3. Click on **Standard.EXE** in the New Project window.
4. Click **OK**.
5. Several blank windows appear, including these: Project1 - Form1(Form) window, Project - Project1 window, and Properties - Form1 window. You can use these windows to create a new project.
6. In the **Properties - Form1** window, change the value of the (Name) field to Example1. In the **Caption** box, enter Wonderful!
7. Choose **File, Save Project as**. You are prompted to save the form with the extension of **.frm**. Enter Example1.frm in the entry area.
8. Microsoft Visual Studio asks if you want to save the Microsoft Visual Basic project file. Specify the name Example1.prj, and click **OK**.
9. Microsoft Visual Studio asks if you want to save it to SourceSafe. Click **No**.
10. Choose **File, Exit**.

You now have a working Microsoft Visual Basic project.

Adding code to your Microsoft Visual Basic Project

The Example1.prj project will compile and run, and present an empty dialog box.

This section describes how to add code to this project to make a very simple IBM Cognos TM1 application.

Procedure

1. Open the Example1.prj project from Windows Explorer.
2. Open the project window, if it is not open already, by choosing **View, Project Explorer**.
3. In the project window, highlight **Project1 (Example.vbp)**. Right-click, and choose **Add, Module**.
4. Click the **Existing** tab, and use the file search dialog to find TM1API.BAS.
5. Double-click the TM1API.BAS file to add it to your project.
6. Choose **View, Code** from the Microsoft Visual Basic menus. A code window appears.
7. Enter the following code in this window:

```
Option Explicit
```

This line forces your variables to be strictly typed by Microsoft Visual Basic. It sometimes helps to make debugging easier.

8. Click the option button containing the word (General). Choose **Form**. The Form_Load subroutine opens.
9. Enter the following code into the Form_Load subroutine:

```
Dim hUser As Long
TM1ApiInitialize()
hUser = TM1SystemOpen()
TM1SystemAdminHostSet hUser, "AdminHost"
```

This code initializes the API, and calls TM1SystemAdminHostSet. Both of these functions must be called in the order shown in any TM1 API program.

Note: The server name established in the second argument to TM1SystemAdminHostSet will have to be a TM1 Admin Host in your network. The name AdminHost is only an example.

If your program now compiles and runs successfully, your project is properly configured. You are ready to write a program that logs into a TM1 server. The following section describes that code.

Logging in to an IBM Cognos TM1 Server

To log in to an IBM Cognos TM1 server from your program, you need to include all of the code shown in the preceding section.

You also must do the following steps:

Procedure

1. Create a TM1 value pool handle with TM1ValPoolCreate. The syntax of this function follows:
`hPool = TM1ValPoolCreate(hUser)`
2. Establish three strings with a valid server name, client name, and password.
`ServerName = "sales" ClientName = "admin" Password = "apple"`
3. Pass the three strings to TM1ValString to create TM1 value capsules from them.
`vServerName = TM1ValString(pPool, Trim(ServerName), 0) vClientName = TM1ValString(pPool, Trim(ClientName), 0) vPassword = TM1ValString(pPool, Trim(Password), 0)`

Note: The strings are trimmed of excess characters before you turn them into value capsules. This is extremely important. Failure to trim these strings will prevent you from logging in successfully.

4. Pass the three TM1 value capsules to the function TM1SystemServerConnect. This function logs you in to the TM1 server.

The following code shows the complete login sequence:

```
Option Explicit
Private Sub Form_Load()
Dim hUser As Long
Dim hPool As Long
Dim hServer As Long
Dim vServerName As Long
Dim vClientName As Long
Dim vClientPassword As Long
Dim ServerName As String * 75
Dim ClientName As String * 75
Dim ClientPassword As String * 75
Dim ErrorMessage As String * 75
' initialize the API
TM1APIInitialize()
hUser = TM1SystemOpen()
' Set the Admin Host Server Name
TM1SystemAdminHostSet hUser, "briant_pc"
' Create a Pool Handle
hPool = TM1ValPoolCreate(hUser)
' Establish Login information
ServerName = "sales"
ClientName = "admin"
ClientPassword = "apple"
vServerName = TM1ValString(hPool, Trim(ServerName), 0)
vClientName = TM1ValString(hPool, Trim(ClientName), 0)
vClientPassword = TM1ValString(hPool, Trim(ClientPassword), 0)
' Log in to a TM1 Server
hServer = TM1SystemServerConnect(hPool, vServerName,
```



```

vClientName, vClientPassword)
' Check to see if we were successful...
If (TM1ValType(hUser, hServer) = TM1ValTypeObject())
Then MsgBox "You Logged in Successfully"
If (TM1ValType(hUser, hServer) = TM1ValTypeError()) Then
MsgBox "The server handle contains an error code."
TM1ValErrorString_VB hUser, hServer, ErrorString, 0
TM1ApiFinalize()
End Sub

```

Logging out of IBM Cognos TM1

The last things to do are to logout from the server and end the API session.

To logout from the server, use `TM1SystemServerDisconnect` as follows:

```
hDisconnect = TM1SystemServerDisconnect (pPool, vServerName)
```

Terminate the API session by calling `TM1SystemClose` and `TM1APIFinalize`.

```

TM1SystemClose (hUser)
TM1APIFinalize()

```

The complete example of the code should now look like the following:

```

Option Explicit
Private Sub Form_Load()
Dim hUser As Long
Dim hPool As Long
Dim hServer As Long
Dim vServerName As Long
Dim vClientName As Long
Dim vClientPassword As Long
Dim ServerName As String * 75
Dim ClientName As String * 75
Dim ClientPassword As String * 75
Dim ErrorString As String * 75
' initialize the API
TM1APIInitialize()
hUser = TM1SystemOpen()
' Set the Admin Host Server Name
TM1SystemAdminHostSet hUser, "briant_pc"
' Create a Pool Handle
hPool = TM1ValPoolCreate(hUser)
' Establish Login information
ServerName = "sales"
ClientName = "admin"
ClientPassword = "apple"
vServerName = TM1ValString(hPool, Trim(ServerName), 0)
vClientName = TM1ValString(hPool, Trim(ClientName), 0)
vClientPassword = TM1ValString(hPool, Trim(ClientPassword), 0)
' Log in to a TM1 Server
hServer = TM1SystemServerConnect(hPool, vServerName,
vClientName, vClientPassword)
If (TM1ValType(hUser, hServer) = TM1ValTypeObject()) Then MsgBox
"The server handle is an object."
If (TM1ValType(hUser, hServer) = TM1ValTypeError()) Then MsgBox
"The server handle is an error code."
TM1ValErrorString_VB hUser, hServer, ErrorString, 0
hPool = TM1SystemServerDisconnect(hPool, vServerName)
TM1SystemClose (hUser)
TM1APIFinalize()
End Sub

```

Chapter 3. IBM Cognos TM1 objects

This section describes the parts of the IBM Cognos TM1 product that you can manipulate using the TM1 API.

The first part of this section is a general description of TM1 objects. Then, each of the objects in the TM1 server is described in its own section.

Example code for some of these objects is contained in files on the TM1 distribution CD.

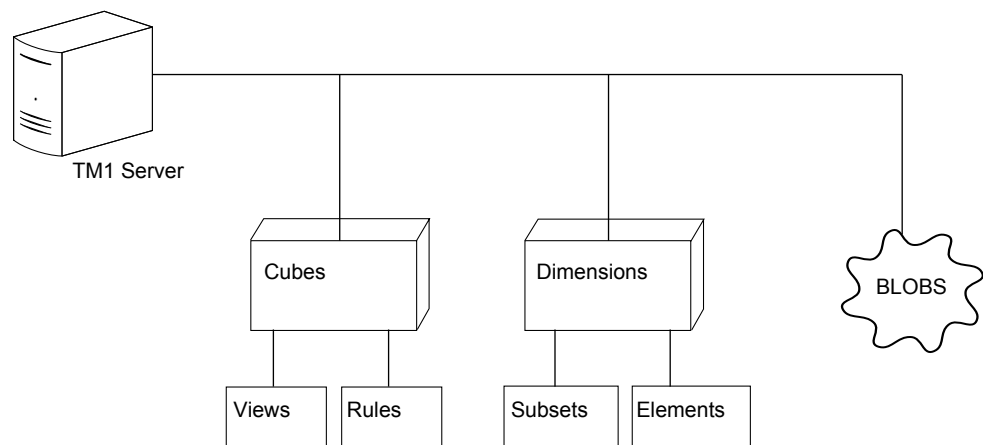
IBM Cognos TM1 objects overview

The IBM Cognos TM1 server and API have an object-oriented architecture.

This simplifies the use of the API, since there are functions that perform common operations on all objects.

The following diagram shows the major objects supported by the API.

Some objects, such as clients and groups, are not reflected in this diagram.



Object handles

Objects are identified and manipulated using object handles.

An object handle is a data structure that determines:

- The server where the object resides.
- The type of object.
- The specific identity of the object. This allows the server to tell, for example, which dimension on the server the object handle refers to.
- The user's read and write access to the object.

Information about an object handle can be extracted using the `TM1ValObject` functions. Object handles remain valid throughout the connection, unless you erase the object, or you destroy the value pool in which the object handle is created. In that case, accessing the object handle results in an error.

Object properties

Objects have a set of properties, which are values associated with the object.

Some properties, such as the type of an object, are defined for all objects. Some others are specific to a type of object.

- Properties of an object can be retrieved using the function `TM1ObjectPropertyGet`.
- Some properties can also be updated using the function `TM1ObjectPropertySet`.

These functions use a value pool, the handle of the object, and a property identifier as arguments. Property identifiers are returned by special functions in the API. For example, to get the name of an object, you call `TM1ObjectPropertyGet`, as follows:

```
TM1ObjectPropertyGet ( hPool, hObject, TM1ObjectName());
```

The `TM1ObjectName` function returns a property index. That index is used to specify which property you are trying to retrieve.

Properties that apply to all (or most) objects start with the prefix `TM1Object`. The following table shows properties that are defined for most objects.

Property	Description
<code>TM1ObjectList</code>	An array of object handles containing a list of all the children of the object of a particular type. Some objects, such as dimensions, have more than one list object associated with them. For example, dimensions have a list for subsets, and a list for hierarchies.
<code>TM1ObjectAttributes</code>	An array of attribute handles containing a list of all of the object's attributes.
<code>TM1ObjectLastTimeUpdated</code>	A <code>TM1V</code> containing a string. This property contains a string indicating the last time the data for this object was updated.
<code>TM1ObjectMemoryUsed</code>	The number of bytes of memory used by the object.
<code>TM1ObjectName</code>	A <code>TM1V</code> containing a string. This property contains the object's name.
<code>TM1ObjectParent</code>	The handle of the object's parent object.

Property	Description
TM1ObjectSecurityStatus	A TM1V containing an integer. This property contains one of the following values, as defined by the API: TM1SecurityRightNone(void); TM1SecurityRightRead(void); TM1SecurityRightWrite(void); TM1SecurityRightReserve(void); TM1SecurityRightLock(void); TM1SecurityRightAdmin(void);
TM1ObjectSecurityOwner	A TM1V containing a string. This property contains the name of the client that owns the security status of the object.
TM1ObjectType	An integer indicating the type of object. The TM1 object type values are defined using special functions in the TM1 API. For example, the TM1 object type for a cube is returned by the function TM1TypeCube().
TM1ObjectReplication	If this object is replicated on a server, this property contains a handle to a replication object.
TM1ObjectReplicationSourceName	The name of the source object in the star server for this object. This property is populated only if the object is replicated.
TM1ObjectRegistration	A TM1V containing an integer. This property contains one of three values, as defined by the API: TM1ObjectPublic(void); TM1ObjectPrivate(void); TM1ObjectUnregistered(void);

Properties for specific objects start with the prefix TM1Cube, TM1Dimension, and so on.

Lists of objects

All user-created objects have a parent object.

The property TM1ObjectParent is used to identify an object's parent.

The parent object has a *list property* that lists all of its children of a given type. For example, server objects have the properties TM1ServerDimensions and TM1ServerCubes. These properties list all the dimensions and cubes in the server.

Views and subsets can be either public or private. IBM Cognos TM1 distinguishes between them both by the property `TM1ObjectRegistration`, and by keeping the objects on different lists. For example, to get a list of public subsets, you need the subset handle, and the `TM1ObjectList...` functions. To get the list of private subsets, you need the `TM1ObjectPrivateList...` functions.

The following functions allow the user to navigate through all the objects in an object list:

- `TM1ObjectListCountGet` returns the number of items on a list.
- `TM1ObjectListHandleByIndexGet` accepts an index argument, and returns the handle of the object in that position in the list. Indexes are one-based. If the index is too large or too small for the number of objects in the list, an error is returned.
- `TM1ObjectListHandleByNameGet` accepts the name of an object on the list as an argument, and returns a handle to it. If no object of the given name can be found in the list, an error is returned.
- `_Ref411736492TM1ObjectListPrivateHandleByIndexGet` accepts an index argument, and returns the handle of an object in that position in the private list. Indexes are one-based. If the index is too large or too small for the number of objects in the list, an error is returned.
- `TM1ObjectListPrivateHandleByNameGet` accepts the name of a object on the private list as an argument, and returns a handle to it. If no object of the given name can be found in the list, an error is returned.

The public and private lists contain only registered objects. It is your responsibility to keep track of unregistered objects. See “Creating and registering objects” for more information on registered and unregistered objects.

Creating and registering objects

You can create new IBM Cognos TM1 objects.

The creation functions are defined individually for each object. See “Unregistered objects” on page 31.

Registering an object inserts it in the public or private *object list property* of its parent. Use the function `TM1ObjectRegister` to register a public object, and the function `TM1ObjectPrivateRegister` to register a private object.

Upon creation, cubes, views, dimensions, and subsets are unregistered by default. Registering an unregistered object does the following:

- Lists it by its handle as a child of its parent object. The parent list property containing the object list handles is used by TM1 to build an object hierarchy.
- Gives the object a name.
- Makes the object persist between client sessions.

Registered objects are also tracked through the TM1 server log file. Changes to the object are stored in server memory, and are recorded in the log file. If the server goes down unexpectedly, you can recover your data by processing the changes in the log file against the copy of the object that is stored on the hard disk. For more information on log files, see “Backup and recovery” on page 18.

If the server is brought down, registered objects are saved to the server disk. Unregistered objects are lost.

Registration functions

You can register objects with functions in IBM Cognos TM1.

TM1 provides the following functions for registering objects:

- `TM1ObjectRegister` registers an object as a public object.
See “`TM1ObjectRegister`” on page 130.
- `TM1ObjectPrivateRegister` registers a private object. You can only access a client object if you created it.
See “`TM1ObjectPrivateRegister`” on page 127.

Note: In order for TM1 to locate the parent object, you must pass `TM1ObjectParent( )` as an argument to one of these functions.

Public, private, and unregistered objects

Public objects are available to other applications and users. They are named and have persistence across client sessions.

Private objects are available only to the IBM Cognos TM1 client that created them. They are named objects and have persistence across client sessions.

Unregistered objects are created during the client session through special `TM1Create` calls. They are unnamed, private, and do not persist across client sessions.

Any TM1 object can be a public object. You register it for public use, by calling the function `TM1ObjectRegister`.

Public objects are accessed by other applications through the `TM1ObjectListHandle` functions. For example, to get a list of dimensions that reside on a server, you can call `TM1ObjectListCountGet` for the TM1 server to determine how many dimensions there are, and then call `TM1ObjectListHandleByIndexGet` to retrieve a list of dimension handles.

Private objects

Currently, views and subsets can be registered as private objects using the function `TM1ObjectPrivateRegister`.

Private objects are accessible only by the user who registered them. A private registered object is a named object. It persists after the termination of a client session.

You can turn a private object into a public object by calling `TM1ObjectPrivatePublish`. When you publish a private view, you must be sure that all of the subsets that comprise that view are public. If any of the subsets that make up the view are private, `TM1ObjectPrivatePublish` will fail with the error `TM1ErrorViewHasPrivateSubsets`.

Unregistered objects

Unregistered objects are intended to be temporary objects.

Suppose you want to browse a specific view of a cube in the client screen. You can create a temporary view and temporary subsets for the dimensions of that cube, display the view and then destroy all those unregistered objects.

When the client disconnects, any unregistered objects you create during the session are destroyed by the server. These objects are not saved on disk when the server goes down.

If you do not explicitly delete unregistered objects using the function `TM1ObjectDestroy`, they are deleted when you call `TM1SystemClose`, or when you destroy the value pool in which the handle to that unregistered object was created.

The functions that return a handle to an unregistered object are as follows:

- `TM1CubeCreate`
- `TM1CubePerspectiveCreate`
- `TM1DimensionCreate`
- `TM1BlobCreate`
- `TM1DimensionCreateEmpty`
- `TM1ViewCreate`
- `TM1SubsetCreateEmpty`
- `TM1RuleCreateEmpty`

Accessing parent objects - security

To create an object or publish a private object, you must have ADMIN permissions to the object's parent.

For example, to create a view (or publish a private view) you must have ADMIN permissions to the parent cube.

If the parent object is a private object you have ADMIN permissions by default. If the parent object is a public object you will need to acquire ADMIN permissions.

To modify an object, you must have WRITE permissions to the object itself. For example, to modify a cube's data, you must have WRITE permissions to the cube.

Note: Modifying meta-data is different from modifying data. Modifying meta-data, such as the structure of a cube (by adding, removing or re-ordering dimensions) is effectively the same as creating a new cube. You must have ADMIN permissions to the server in order to do this.

To delete an object, you must have ADMIN permissions to the object's parent. For example, to delete a public subset, you must have ADMIN permissions to the dimension.

For information on security, refer to the *Integra Security Guide*.

Object names

Object names are strings that should be no more than 128 bytes in length.

IBM Cognos TM1 is insensitive to case or embedded blanks in an object name. For example, if there is an object with the name 'Abc Def', that object can be referred to using the names 'ABCDEF', 'ab cd ef' etc. When the file name exceeds the limit of the operating system, the error `TM1ErrorObjectNameTooLong` is returned.

The following TM1 objects of the same registration (public or private) must have unique names:

- Cubes

- Dimensions
- Clients
- Groups
- BLOBs

Elements must be unique within a dimension and cannot have duplicated names. For example, you cannot have two states named New York in your States dimension.

You can have two views with the same name, as long as one of the views is private, and one is public. You can also have two subsets with the same name, as long as one of the subsets is private, and one is public.

You cannot have more than one public subset, or more than one public view with the same name. Similarly, a single user cannot have more than one private subset, or more than one private view with the same name.

For more details, see the section, "Understanding TM1 Object Naming Conventions", in the IBM Cognos TM1 *Developers Guide*.

Loading objects

When an IBM Cognos TM1 server is loaded, it automatically builds lists of available objects and registers them.

These lists are based on the files that reside in the data directories provided to the server. The objects themselves are not loaded until they are needed.

Procedure

1. Call `TM1ObjectFileLoad` to load the file.
2. Call `TM1ObjectRegister` to include the object name and handle in the corresponding object list of the parent.
3. Newly registered objects are saved to the first directory in the data directory list of the server.

Deleting objects

The function you use to delete an object depends on whether or not the object has been registered, and whether the object is public or private.

Use `TM1ObjectDestroy` to delete any object that has been created, but not registered. This function removes the object from memory and releases its space.

You should take care to destroy any unregistered object that is no longer needed by the application. Unregistered objects take up memory, and can slow down your system if you allow them to persist.

If you have registered the object as a public object, you call `TM1ObjectDelete` to delete the object from its parent's object list, remove it from memory, and release its space.

If you have registered the object as a private object, you call `TM1ObjectPrivateDelete` to delete the object from its parent's object list, remove it from memory, and release its space.

Also, when deleting an object, you may want to delete the file from where it was loaded using the function `TM1ObjectFileDelete`. Files for deleted objects are automatically deleted when the server is shut down, but you can free up disk space as soon as possible by deleting the files yourself.

Saving objects to disk

An object, registered or unregistered, can be saved to a file by using the function `TM1ObjectFileSave`.

Once an object is registered, it is saved to disk automatically when the server is brought down.

Copying objects

Use `TM1ObjectDuplicate` to make a copy of an object.

`TM1ObjectDuplicate` makes a memory copy of the target object. The resulting object is unregistered, and resides on the same server.

Use `TM1ObjectCopy` to make a copy of an object that resides on another server. `TM1ObjectCopy` copies an object on one server to an empty handle of an object of the same type on another server. The resulting object is unregistered.

Server object

An IBM Cognos TM1 server contains cubes, which, in turn are defined by one or more dimensions.

The server loads data files from disk, and creates a set of memory objects that can be accessed and updated using the API.

A server object in the API corresponds to a connection to a running TM1 server. The user creates a server handle by calling the `TM1SystemServerConnect` function. The handle is destroyed by the `TM1SystemServerDisconnect` function.

Server properties

To access all other objects in a server, look at the properties of the server.

The following properties contain lists of objects registered in the server:

Property	Description
<code>TM1ServerCubes</code>	A list of cube handles. This list contains every cube registered with the server.
<code>TM1ServerDimensions</code>	A list of dimension handles. This list contains every dimension registered with the server.
<code>TM1ServerGroups</code>	A list of group handles. This list contains every group established on the server.
<code>TM1ServerClients</code>	A list of client handles. This is a list of every client connected to the server.

Use the TM1ObjectList functions to navigate through these lists. The following standard object properties are defined for servers:

- TM1ObjectMemoryUsed
- TM1ObjectName
- TM1ObjectType

Server functions

You can use several IBM Cognos TM1 API server functions.

The following table lists the TM1 API server functions:

Function	Purpose
TM1SystemServerClientName	Returns the name of a client logged in to a server. See “TM1SystemServerClientName” on page 181.
TM1SystemServerConnect	Logs in a client. See “TM1SystemServerConnect” on page 182.
TM1SystemServerDisconnect	Logs out a client. See “TM1SystemServerDisconnect” on page 183.
TM1SystemServerHandle	Returns the handle to a server object. See “TM1SystemServerHandle” on page 184.
TM1SystemServerName	Returns the name of a server, given its position in the Tm1.adm file. See “TM1SystemServerName” on page 185.
TM1SystemServerName_VB	Returns the name of a server in a Visual Basic application. See “TM1SystemServerName_VB” on page 327.
TM1SystemServerNof	Returns the number of servers in the Tm1.adm file. See “TM1SystemServerNof” on page 185.
TM1SystemServerStart	Starts the local TM1 server. See “TM1SystemServerStart” on page 186.
TM1SystemServerStop	Stops the local TM1 server. See “TM1SystemServerStop” on page 188.

Dimension and element objects

A dimension is a list of related items, called elements, which you would find listed down the rows, or across the columns of a report.

For example, the dimension Region might contain elements France, Germany and United States. Dimensions are used to construct cubes, and serve as indexes to the data in a cube.

Parent object

The parent of a dimension object is the IBM Cognos TM1 server.

The TM1 server has a list property `TM1ServerDimensions` that contains a list of every dimension registered with the server.

Child objects

Elements and consolidated elements are children of dimensions.

These are listed in the property `TM1DimensionElements`.

A dimension can have named subsets, which list one or more elements that belong to the dimension. Subsets are children of a dimension object. A dimension's subsets are listed in the property `TM1DimensionSubsets`. Note that subsets can be public or private objects.

Hierarchies are also child objects of a dimension. Hierarchies are multi-leveled, ordered groups of elements. A dimension's hierarchies are listed in the property `TM1DimensionHierarchies`.

Registration

To register a public dimension, call `TM1ObjectRegister`.

Dimensions cannot be registered as private objects in the current release of the IBM Cognos TM1 API.

Creating a dimension

You can create new dimensions.

The following steps illustrate how to create a new dimension.

Procedure

1. Call `TM1DimensionCreateEmpty`. This function returns a handle to an empty dimension.
2. Populate the dimension with simple elements by calling `TM1DimensionElementInsert`. You add consolidated elements by calling `TM1DimensionElementComponentAdd`.
3. Once the dimension has been populated, call `TM1DimensionCheck` to verify the integrity of the new dimension.
4. If the integrity is intact, register the dimension with `TM1ObjectRegister`.

Updating a dimension

You can update dimensions.

The following steps illustrate how to update an existing dimension.

Procedure

1. Call `TM1ObjectDuplicate` to make a copy of the dimension you want to update. The copy is an unregistered dimension.
2. Use the calls `TM1DimensionElementInsert` and `TM1DimensionElementComponentAdd` to add new elements and components to the duplicated, unregistered dimension.
3. Use the calls `TM1DimensionElementDelete` and `TM1DimensionElementComponentDelete` to delete unneeded elements and components.
4. Call `TM1DimensionCheck` to verify the integrity of the new dimension.
5. Call `TM1DimensionUpdate` to overwrite the old dimension with the new one.

Deleting a dimension

To delete a public dimension, call `TM1ObjectDelete`.

Unregistered dimensions are deleted with the function `TM1ObjectDestroy`, or are deleted when the API session ends with `TM1SystemClose`.

Elements

The elements of a dimension are arranged in consolidation hierarchies.

At the bottom of a hierarchy (level 0) are the simple or leaf elements. These elements, in turn can be combined into first level consolidated elements. These consolidated elements can in turn be combined into higher level consolidations.

For example, a dimension depicting geography might start with City as the bottom or input level. Cities could then be consolidated into Counties, Counties into States, States into Regions, and so on.

IBM Cognos TM1 classifies elements according to the following types:

Type	Description
Simple	Simple elements are used to identify cells where numbers may be entered and stored. The level of a simple element is always zero.
Consolidated	Consolidated elements are defined as a weighted sum of component elements. The component elements may be simple or consolidated. The level of a consolidated element is the highest of the levels of its components plus one.
String	TM1 supports cubes that contain string information. String elements are used to identify such cells. String elements cannot be used in consolidations.

Dimension properties

In addition to all standard object properties, dimensions have properties.

The following properties are available:

Property	Description
TM1DimensionElements	A list of element handles containing all the elements in the dimension.
TM1DimensionSubsets	A list of subsets registered with dimension.
TM1DimensionNofLevels	An IBM Cognos TM1V containing an integer indicating number of levels in dimension. For example, a dimension contains 3 levels. The simple elements (Jan, Feb, Mar, and so on) are level 0 elements. The Year consolidation is at level 2.
TM1DimensionWidth	A TM1V containing an integer. The number is the width of the widest element in the dimension, in characters. This is useful for setting the width of a spreadsheet column.
TM1DimensionTopElement	An element object handle. This element is the top-level consolidation of the dimension, and can be used as a starting point for drilling down within the dimension.

Dimension functions

You can create, delete, and manipulate dimension objects.

The following functions allow you to create, delete, and manipulate dimension objects:

Function	Purpose
TM1DimensionCheck	Checks a dimension for consistency. See “TM1DimensionCheck” on page 97.
TM1DimensionCreateEmpty	Returns a handle to an empty dimension. See “TM1DimensionCreateEmpty” on page 96.
TM1DimensionElementComponentAdd	Adds a component to a consolidated element. See “TM1DimensionElementComponentAdd” on page 98.
TM1DimensionElementComponentDelete	Deletes a component from a consolidated element. See “TM1DimensionElementComponentDelete” on page 99.

Function	Purpose
TM1DimensionElementComponentWeightGet	Returns the weight of a component in a consolidated element. See “TM1DimensionElementComponentWeightGet” on page 100.
TM1DimensionElementDelete	Deletes an element from a dimension. See “TM1DimensionElementDelete” on page 101.
TM1DimensionElementInsert	Inserts an element into a dimension. See “TM1DimensionElementInsert” on page 101.
TM1DimensionUpdate	Replaces a registered dimension with a new one and updates all associated cubes. See “TM1DimensionUpdate” on page 108.

Cube objects

Cubes are the main repository of data in an IBM Cognos TM1 server.

Cubes are structured by combining two to sixteen dimensions into a data matrix. Each value or cell in the cube is identified using one element from each of its dimensions. For example, a cube made up of dimensions Region, Account and Month may contain a cell identified as Northeast, Sales, and January.

A cell in a cube can accept input if all its identifying elements are of the simple type. A cell can contain string values if any of its identifying elements is of type string. Otherwise, the cell is consolidated and cannot accept input, or its value is established by a rule.

Parent object

The parent of a cube object is the IBM Cognos TM1 server.

The TM1 server has a list property `TM1ServerCubes` that contains a list of every cube registered with the server.

Child objects

Cubes can contain rules and views.

The rule for a cube is contained in the cube property `TM1CubeRule`. The property `TM1CubeViews` contains a list of all the named views of the cube. Views can be public or private objects.

Registration

To register a cube as a public object, call `TM1ObjectRegister`.

Cubes cannot be registered as private objects in the current release of the IBM Cognos TM1 API.

Creating a cube

To create a new cube handle, call `TM1CubeCreate`.

This function returns a handle to an empty cube.

Retrieving and updating cube cells

The function `TM1CubeCellValueGet` retrieves the value of a cell in a cube.

The function `TM1CubeCellValueSet` updates the value of a cell in a cube.

Deleting a cube

To delete a cube, call `TM1ObjectDelete`.

Unregistered cubes are deleted when you call the function `TM1ObjectDestroy`, or when you end the API session by calling `TM1SystemClose`.

Cube properties

A cube object has properties.

The following properties are available:

Property	Description
<code>TM1CubeDimensions</code>	Contains an array of dimension handles. This is a list of the dimensions in a cube. This is an ordered list.
<code>TM1CubeRule</code>	Contains the handle to the rule that applies to a cube. Dimensions may be used to construct many different cubes, but rules are specific to a particular cube. Note that this property can be NULL. A cube does not need to have a rule.
<code>TM1CubeViews</code>	Contains a list of view handles. This list contains all the views associated with the cube. Call <code>TM1ObjectPrivateListHandleByIndexGet</code> to retrieve a private view. Call <code>TM1ObjectListHandleByIndexGet</code> to retrieve a public view.
<code>TM1CubePerspectiveMemory</code>	Contains the maximum number of bytes that can be used to store perspectives. If this value is set to 0, no perspectives will be stored permanently for this cube.
<code>TM1CubePerspectiveMinimumTime</code>	Contains the number of seconds required to calculate a perspective, below which the perspective will not be stored. The recommended value is 5. The default for this property is 5.

Cube functions

An IBM Cognos TM1 cube has functions.

The following table lists the TM1 cube functions.

Function	Description
TM1CubeCellValueGet	Retrieves the value from the cell of a cube. See “TM1CubeCellValueGet” on page 83.
TM1CubeCellValueSet	Sets the value from the cell of a cube. See “TM1CubeCellValueSet” on page 84.
TM1CubeCreate	Creates a new cube. See “TM1CubeCreate” on page 86.
TM1CubePerspectiveCreate	Create a cube perspective. See “TM1CubePerspectiveCreate” on page 88.
TM1CubePerspectiveDestroy	Delete a cube perspective. See “TM1CubePerspectiveDestroy” on page 90.

Cube perspectives

A perspective is a sub-cube of a Cube.

For example, suppose a cube has four dimensions: Products, Regions, Accounts and Months. If you want to look at Accounts by Months for the Total of Products and the total of Regions, you can construct a perspective that has two dimensions: Accounts and Months. This perspective contains the desired values.

Perspectives are used to improve performance. Calculating a perspective takes much less time than calculating all the cells in the sub-cube individually. Therefore, using a perspective with simple, string, or low-level-consolidation elements is not recommended.

Once a perspective is created, any time a value is requested of the Cube that exists in the perspective, it is retrieved from the perspective rather than calculated from the Cube.

- Use the function TM1CubePerspectiveCreate to create a perspective. Use the function TM1CubePerspectiveDestroy to destroy a perspective.

You should always destroy perspectives when you finished using them. IBM Cognos TM1 will make a judgment, based on the time it took to compute the perspective and available memory, on whether to retain it or clear it from memory.

The criteria for storing perspectives is controlled by the following properties of the Cube:

- `TM1CubePerspectiveMaximumMemory` sets the maximum number of bytes that can be used to store perspectives. If this property is set to zero, no perspectives will be stored permanently.
- `TM1CubePerspectiveMinimumTime` sets the number of seconds required to calculate a perspective below which the perspective will not be stored. The default and recommended value is 5 seconds.

Rule objects

Cube rules expand the standard hierarchical consolidation operations that you may define within a dimension.

For most applications, these simple consolidations are enough to cover the bulk of the calculations required.

For more complex calculations, such as multiplication, division, financial functions, and so on, IBM Cognos TM1 provides the cube rules multidimensional language. Please refer to other TM1 documents for details on how to write rules.

Rules are written as a series of statements, which are permanently stored as an ASCII file. When the rule is loaded, it is stored in memory in its original form. The rule statements are checked for syntax and consistency, and translated into a compact form for execution.

Rule properties

The only property defined for rules is `TM1RulesNofLines`, which contains the number of lines of text in the rule.

To retrieve the line of text, use the function `TM1RuleLineGet`. The string returned by `TM1RuleLineGet` can be up to 64k bytes.

Rule functions

The following table lists the IBM Cognos TM1 rule functions.

Function	Description
<code>TM1RuleAttach</code>	Attaches a rule to a cube. See “ <code>TM1RuleAttach</code> ” on page 139.
<code>TM1RuleCheck</code>	Checks the syntax of a rule. See “ <code>TM1RuleCheck</code> ” on page 139.
<code>TM1RuleCreateEmpty</code>	Creates an empty rule, and returns a handle to that rule. See “ <code>TM1RuleCreateEmpty</code> ” on page 140.
<code>TM1RuleDetach</code>	Detaches a rule from a cube. See “ <code>TM1RuleDetach</code> ” on page 141.

Function	Description
TM1RuleLineGet	Retrieves a line from a rule. See “TM1RuleLineGet” on page 141.
TM1RuleLineInsert	Inserts a line in a rule. See “TM1RuleLineInsert” on page 142.
TM1RuleLineDelete	Deletes a line from a rule.

Creating a new rule

The following steps illustrate how to create a rule.

Procedure

1. Create an empty Rule using the function TM1RuleCreateEmpty.
TM1RuleCreateEmpty returns a handle to an empty rule.
2. Using the handle to the Rule, insert new statements using the function TM1RuleLineInsert.
3. After all the statements are in, check that the Rule is properly defined using the function TM1RuleCheck.
4. To enable the Rule, attach it to a Cube using the TM1RuleAttach.

Updating an existing rule

The following steps illustrate how to update an existing rule.

Procedure

1. Use the TM1ObjectDuplicate function to create a copy of the rule.
2. Use the functions TM1RuleLineInsert and TM1RuleLineDelete to add and delete lines from the rule.
3. Check that the rule is properly defined using the TM1RuleCheck function.
4. Replace the existing rule using the TM1RuleAttach function. Note that this function automatically destroys the old rule.
5. To remove an existing rule without attaching a new one, use the function TM1RuleDetach.

Subset objects

A subset is an ordered set of elements selected from a dimension.

A subset can be all the elements of a dimension, a smaller set selected by the user, or a larger set containing duplicate elements.

Parent object

The parent of a subset object is a dimension.

The dimension property TM1DimensionSubsets contains handles to all of the named subsets in the dimension.

Registration

To register a public subset with its parent cube, call `TM1ObjectRegister`.

You need at Admin privileges to the parent dimension to create public subsets.

To register a private subset, call `TM1ObjectPrivateRegister`.

Creating a subset

To create a new subset handle, call `TM1SubsetCreateEmpty`. This function returns a handle to an empty subset.

You can add elements to an empty subset with the functions `TM1SubsetAll` and `TM1SubsetInsertElement`.

Deleting a subset

To delete a public subset, call `TM1ObjectDelete`. To delete a private subset, call `TM1ObjectPrivateDelete`.

To delete an unregistered subset, call the function `TM1ObjectDestroy`, or end the API session by calling `TM1SystemClose`.

Updating a subset

The following steps illustrate how to update an existing subset.

Procedure

1. Call `TM1ObjectDuplicate` to make a copy of the subset you want to update. This returns a handle to an unregistered subset.
2. Use the `TM1Subset` functions to add and delete elements to the unregistered subset.
3. Call `TM1SubsetUpdate` to overwrite the old subset with the new one.

Subset object properties

A subset object has the following properties.

Property	Description
<code>TM1SubsetsElements</code>	A list of element handles. These are the elements that make up the subset. Use this property only with the function <code>TM1ObjectListHandleByIndexGet</code> . If you use this property with <code>TM1ObjectListHandleByNameGet</code> , you will receive an error.
<code>TM1SubsetAlias</code>	A string. If the string is NULL, no alias has been applied to the subset. Otherwise, this string is the name of the alias attribute to be used when displaying element names in this subset.

Subset functions

The following table lists the IBM Cognos TM1 subset functions.

Function	Description
TM1SubsetsAll	Populates a subset with all elements of the parent dimension. See “TM1SubsetAll” on page 155.
TM1SubsetCreateEmpty	Creates an empty subset object. See “TM1SubsetCreateEmpty” on page 157.
TM1SubsetElementDisplay	Returns a string containing encoded display information for an element. See “TM1SubsetElementDisplay” on page 157.
TM1SubsetInsertElement	Inserts an element into a subset. See “TM1SubsetInsertElement” on page 165.
TM1SubsetInsertSubset	Inserts one subset into another. See “TM1SubsetInsertSubset” on page 165.
TM1SubsetSelectByAttribute	Selects or de-selects elements of a subset that have attributes values that match a given value. See “TM1SubsetSelectByAttribute” on page 168.
TM1SubsetSelectByIndex	Selects or de-selects an element of a subset by its index. This does not change the contents of the subset. It just toggles the selection flag on the element. See “TM1SubsetSelectByIndex” on page 169.
TM1SubsetSelectByLevel	Selects or de-selects all elements of a given level. This does not change the contents of the subset. It just toggles the selection flag on the element. See “TM1SubsetSelectByLevel” on page 170.
TM1SubsetSelectByPattern	Selects or de-selects all elements whose names match a regular expression. This does not change the contents of the subset. It just toggles the selection flag on the element. See “TM1SubsetSelectByPattern” on page 170.

Function	Description
TM1SubsetSelectionDelete	Deletes selected elements from a subset. This function, unlike the other selection functions, actually changes the contents of the subset. See “TM1SubsetSelectionDelete” on page 171.
TM1SubsetSelectionInsertChildren	Inserts the children of every selected element in the subset. Children are inserted after the parent. See “TM1SubsetSelectionInsertChildren” on page 172.
TM1SubsetSelectionInsertParents	Inserts the parents of every selected element in a subset. Parents are inserted above their children. See “TM1SubsetSelectionInsertParents” on page 173.
TM1SubsetSelectionKeep	Removes all elements from the subset that are not selected. See “TM1SubsetSelectionKeep” on page 173.
TM1SubsetSelectNone	Clears the selection from any selected elements in a subset. See “TM1SubsetSelectNone” on page 174.
TM1SubsetSort	Sorts the elements in a subset alphabetically. See “TM1SubsetSort” on page 174.
TM1SubsetSortByHierarchy	Arranges the elements in a subset by placing all parent and children elements together, with the children grouped under the parents. See “TM1SubsetSortByHierarchy” on page 175.
TM1SubsetSubtract	Removes a set of elements from a subset. See “TM1SubsetSubtract” on page 176.
TM1SubsetUpdate	Updates a registered subset. See “TM1SubsetUpdate” on page 176.

Subset element display functions

The IBM Cognos TM1 API version includes functions to support the development of a sophisticated and elegant graphical user interface.

The functions that support the user interface have the prefix TM1ElementDisplay.

In Perspectives, elements in a subset are displayed in an outline tree. The outline tree is drawn based on dependencies between elements in the subset in their current order.

Each element is displayed indented according to its position in the hierarchy, and the elements are connected by a tree outline structure. Each line in the display contains graphic elements, which connect with those of the previous and following line to present the total tree. Going from right to left, each line is composed of the following elements:

- The name of the element.
- An icon for the element type.
- A connector which can be a tee, an ell, or no connector at all. The connector may also contain a box with a plus sign (+) if the element has children but none are currently displayed directly beneath it, a box with a minus sign (-) if its children are displayed directly beneath it, or no box if the element has no children.
- A series of vertical bars to connect to higher levels in the tree.

The information necessary to draw a line corresponding to a subset element is available through the function TM1SubsetElementDisplay. This function returns an encoded character string. This string is then used as an argument for a series of functions that extract display information for the element. The following table describes the subset element display functions.

Function	Description
TM1SubsetElementDisplayEll	Returns a Boolean. If TRUE, the element connector is an "ell" as with 4 Quarter and Dec in the example above. See "TM1SubsetElementDisplayEll" on page 158.
TM1SubsetElementDisplayPlus	Returns a Boolean. If TRUE, the element connector includes a "plus" box as with 1 Quarter in the example above. See "TM1SubsetElementDisplayPlus" on page 161.
TM1SubsetElementDisplayMinus	Returns a Boolean. If TRUE, the element connector includes a "minus" box as with 2 Quarter and 4 Quarter in the example above. See "TM1SubsetElementDisplayMinus" on page 160.

Function	Description
TM1SubsetElementDisplayLevel	Returns an index indicating the number of levels the element is to be indented in the display. This does not correspond to the level in the dimension hierarchy. This level is zero-based, left-to-right indentation indicator. For example, Year is at level zero, 2 Quarter is at level one, and Jun is at level two. See “TM1SubsetElementDisplayLevel” on page 159.
TM1SubsetElementDisplayLine	Returns a Boolean. If TRUE, the column in the <i>n</i>th position from left to right will contain a vertical line. For example April has a vertical line in position one, and October has no vertical lines. See “TM1SubsetElementDisplayLine” on page 159.

View objects

A view is a user-defined selection of data from a cube.

This data can be displayed in a two-dimensional table, such as a spreadsheet.

Parent object

The parent of a view object is a cube.

The cube property TM1CubeViews contains handles to all of the named views of the cube.

Registration

To register a view with its parent cube, call TM1ObjectRegister.

You need ADMIN privileges to the parent cube to create a public view.

To register a private view, call TM1ObjectPrivateRegister.

Creating a view

To create a new view handle, call TM1ViewCreate.

Deleting a view

You can delete a view in several ways.

To delete a public view, call TM1ObjectDelete. To delete a private view, call TM1ObjectPrivateDelete.

To delete an unregistered view, call TM1ObjectDestroy or end the API session with TM1SystemClose.

View object properties

View object properties determine the characteristics of a view.

Properties for view objects are as follows:

Property	Description
TM1ViewColumnSubsets	An array of subset or dimension handles. These are displayed in the columns of the view. If all of the elements of a dimension are included in the view, the array element will be a dimension handle. If only some of the elements of the dimension are included in the view, the array element will be a subset handle.
TM1ViewPreConstruct	A Boolean. If TRUE, values for the view are calculated when the server initializes.
TM1ViewRowSubsets	An array of subset or dimension handles. These subsets are displayed in the rows of the view. If all of the elements of a dimension are included in the view, the array element will be a dimension handle. If only some of the elements of the dimension are included in the view, the array element will be a subset handle.
TM1ViewSuppressZeroes	A Boolean. If TRUE, rows and columns containing only zeroes are not displayed in the view.
TM1ViewTitleElements	An array of index handles. These indexes correspond to the elements of the subset or dimension that are displayed in the view. The number of handles in this array is always the same as the number of handles in the TM1ViewTitleSubsets array. The items in the two arrays also correspond - The first handle in this array corresponds to the first handle in the TM1ViewTitleSubsets array; the second handle in this array belongs to the second handle in the TM1ViewTitleSubsets array, and so on.
TM1ViewTitleSubsets	An array of subset or dimension handles. These subsets are displayed in the title area of the view. If all of the elements of a dimension are included in the view, the array element will be a dimension handle. If only some of the elements of the dimension are included in the view, the array element will be a subset handle.

Property	Description
TM1ViewFormat	A string. If the string is NULL, this view has no format. The string has the following format: $n:fp$ where f is the format of the cell and p is the precision. The following formats are supported: C = Currency G = General P = Percentage SC = Scientific For example, if the view displays numeric data in currency format, the TM1ViewFormat string would be $n:C2$. The currency symbol is determined by the Microsoft Windows Regional settings in the control panel.
TM1ViewShowAutomatically	A Boolean. If TRUE, the data for the view is automatically re-displayed in Perspectives when the view is re-configured.

View functions

The following table lists the IBM Cognos TM1 view functions.

Function	Description
TM1ViewArrayColumnsNof	Returns the number of columns in a view array. See “TM1ViewArrayColumnsNof” on page 211.
TM1ViewArrayRowsNof	Returns the number of rows in a view array. See “TM1ViewArrayRowsNof” on page 213.
TM1ViewArrayConstruct	Constructs a two-dimensional array of information that can be used to display a view. See “TM1ViewArrayConstruct” on page 212.
TM1ViewArrayDestroy	Destroys a view array. See “TM1ViewArrayDestroy” on page 213.

Function	Description
TM1ViewArrayValueGet	Retrieves a single value from a view. See “TM1ViewArrayValueGet” on page 214.
TM1ViewCreate	Creates a new view object. See “TM1ViewCreate” on page 218.

BLOB objects

A BLOB (Binary Large Objects) is a data file.

You can store, retrieve, and modify BLOBs on the IBM Cognos TM1 server.

Property object

The parent of a BLOB object is the IBM Cognos TM1 server.

The IBM Cognos TM1 server has a list property TM1ServerBlobs that contains a list of every BLOB stored on the server.

Child object

BLOBs have no child objects.

Registration

BLOBs require no registration.

Creating a BLOB

The following steps illustrate how to create a new BLOB.

Procedure

1. Call TM1BlobCreate. This function returns a handle to an empty BLOB.
2. Call TM1BlobPut to add data to the BLOB.
3. Call TM1BlobClose when you are finished adding information to the BLOB.

Updating a BLOB

The following steps illustrate how to update an existing BLOB.

Procedure

1. Call TM1BlobOpen for the blob you want to modify.
2. Use the calls TM1BlobGet and TM1BlobPut to retrieve and modify information in the BLOB.
3. Call TM1BlobClose to write the modified BLOB back to the server.

Deleting a BLOB

Use TM1ObjectDelete to delete a BLOB.

File storage

BLOBs exist as files on the server.

The name of the file is set when you call the function TM1BlobCreate.

BLOB object properties

A BLOB object has one property: TM1BlobSize. This property is a TM1V containing a number.

The number indicates the size of the BLOB, in bytes.

Blob functions

The functions that manipulate BLOBs are listed in the following table.

Function	Purpose
TM1BlobClose	Closes a BLOB. See “TM1BlobClose” on page 56.
TM1BlobOpen	Opens a BLOB for read or write See “TM1BlobOpen” on page 57.
TM1BlobCreate	Creates a new BLOB, and returns a handle to it. See “TM1BlobCreate” on page 56.
TM1BlobGet	Reads data from a BLOB. See “TM1BlobGet” on page 57.
TM1BlobPut	Writes data to a BLOB. See “TM1BlobPut” on page 58.

Chapter 4. IBM Cognos TM1 functions for C

This section contains a complete description of the C functions contained in the IBM Cognos TM1 API.

The functions are presented in alphabetical order.

- Function Types and Naming Conventions
- Function List
- Configuring the TM1 C API to Use SSL

Function types and naming conventions

There are several groups of functions included in the IBM Cognos TM1 API.

The API uses an ordered naming system to separate the functions into groups.

All function names begin with TM1, followed by the object and sub-object to which the function applies, optionally followed at the end by a verb that describes the action taken. For example:

TM1ClientGroupAssign

The API functions can be grouped as follows:

- System functions, which are used to interact with the API itself and do not involve interaction with the servers start with the prefix TM1System.
- Functions that are used to operate on value capsules start with the prefix TM1Val.
- Functions that operate on all objects start with the prefix TM1Object.
- Functions that are used to operate on servers, cubes, dimensions and other objects start with the prefix TM1ObjectType. For example: Server objects start with the prefix TM1Server. Cube objects start with the prefix TM1Cube. Dimension objects start with the prefix TM1Dimension, and so on.

Configuring the IBM Cognos TM1 C API to use SSL

The following public routines are part of the IBM Cognos TM1 C API.

You can use these routines to configure a client to communicate with the Admin Server using SSL.

TM1SystemSetAdminSSLCertAuthority

Use this routine to set the name of the certificate authority that issued the IBM Cognos TM1 Admin Server's certificate.

```
void TM1SystemSetAdminSSLCertAuthority(TM1U hUser, CHAR *  
szAdminSSLCertAuthority)
```

TM1SystemGetAdminSSLCertAuthority

Use this routine to retrieve the name of the certificate authority that issued the IBM Cognos TM1 Admin Server's certificate.

```
CHAR * TM1SystemGetAdminSSLCertAuthority(TM1U hUser)
```

TM1SystemSetAdminSSLCertRevList

Use this routine to set the name of the certificate revocation file, which is issued by the certificate authority that originally issued the IBM Cognos TM1 Admin server's certificate.

```
void TM1SystemSetAdminSSLCertRevList(TM1U hUser, CHAR *  
szAdminSSLCertRevList)
```

TM1SystemGetAdminSSLCertRevList

Use this routine to retrieve the name of the certificate revocation file, which is issued by the certificate authority that originally issued the IBM Cognos TM1 Admin server's certificate.

```
CHAR * TM1SystemGetAdminSSLCertRevList(TM1U hUser)
```

TM1SystemSetAdminSSLCertID

Use this routine to set the name of the principal to whom the IBM Cognos TM1 Admin server's certificate is issued.

```
void TM1SystemSetAdminSSLCertID(TM1U hUser, CHAR * szAdminSSLCertID)
```

TM1SystemGetAdminSSLCertID

Use this routine to retrieve the name of the principal to whom the IBM Cognos TM1 Admin server's certificate is issued.

```
CHAR * TM1SystemGetAdminSSLCertID(TM1U hUser)
```

TM1SystemSetExportAdminSvrSSLCertFlag

Use this routine to set the flag that indicates that the certificate authority certificate, which originally issued the IBM Cognos TM1 Admin server's certificate, should be exported from the certificate store.

```
void TM1SystemSetExportAdminSvrSSLCertFlag(TM1U hUser, TM1_BOOL  
bExportAdminSvrSSLCert)
```

TM1SystemSetAdminSvrExportKeyID

Use this routine to set the identity key used to export the certificate authority certificate, which originally issued the IBM Cognos TM1 Admin server's certificate, from the certificate store.

```
void TM1API TM1SystemSetAdminSvrExportKeyID(TM1U hUser, CHAR *  
szAdminSvrExportKeyID)
```

TM1APIInitialize

Initializes the TM1 API. Call this function at the beginning of your IBM Cognos TM1 application.

Item	Description
Purpose	Initializes the TM1 API. Call this function at the beginning of your IBM Cognos TM1 application.
Definition	TM1IMPORT void TM1API TM1APIInitialize(void);
Parameters	None.
Result	This function performs per-process initialization for the TM1 API. This function was added to avoid some memory conflicts that can occur in very complicated TM1 applications that involve multiple users. You should call this function at the beginning of your TM1 API application. This function is part of the API initialization sequence required by every TM1 API program. See "Connecting to the API" for more information. You should call TM1APIFinalize at the end of your TM1 API application.
Security	None.
Errors	None.
See Also	TM1APIFinalize

TM1APIFinalize

Cleans up memory structures used during IBM Cognos TM1 API processing.

Item	Description
Purpose	Cleans up memory structures used during IBM Cognos TM1 API processing.
Definition	TM1IMPORT TM1V TM1API TM1APIFinalize();
Parameters	None.
Result	This function cleans up memory locks and performs other cleanup for the TM1 API. You should call this function at the end of your TM1 API application. You should call TM1APIInitialize at the beginning of your TM1 API application.
Security	None.
Errors	None.

Item	Description
See Also	TM1APIInitialize

TM1BlobClose

Closes the BLOB. When finished with reading or writing, you should call this function to close the BLOB.

Item	Description
Purpose	Closes the BLOB. When finished with reading or writing, you should call this function to close the BLOB.
Definition	<pre>TM1IMPORT TM1V TM1API TM1BlobClose(TM1P hPool, TM1V hBlob);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hBlob is the handle of the BLOB.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	TM1ErrorBlobCloseFailed
See Also	Other TM1Blob functions.

TM1BlobCreate

Creates a BLOB with the specified name and registers the object on the server. You don't need to make a separate registration call to register the object.

Item	Description
Purpose	Creates a BLOB with the specified name and registers the object on the server. You don't need to make a separate registration call to register the object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1BlobCreate(TM1P hPool, TM1V hServer, TM1V sName);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server on which the BLOB will be created. sName is a TM1_STRING containing the name of the BLOB to be created.

Item	Description
Result	The function returns the handle to the BLOB created.
Security	None.
Errors	TM1ErrorBlobCreateFailed
See Also	Other TM1Blob functions.

TM1BlobGet

Retrieves n bytes of data from the BLOB starting at location x.

Item	Description
Purpose	Retrieves n bytes of data from the BLOB starting at location x. The data is returned in the argument buf. The application is responsible for allocating the memory for the data returned.
Definition	<pre>TM1IMPORT unsigned long TM1API TM1BlobGet(TM1U hUser, TM1V hBlob, TM1_INDEX x, TM1_INDEX n, CHAR *buf);</pre>
Parameters	hUser is the user handle obtained with TM1SystemOpen. hBlob is the handle to the BLOB. x is the starting location in the BLOB to retrieve data from. n is the number of bytes to retrieve buf is the location where retrieved data will be put. Caller is responsible for allocating enough memory to hold the requested data.
Result	Returns the number of bytes successfully read from the BLOB.
Security	None.
Errors	None.
See Also	Other TM1Blob functions.

TM1BlobOpen

Opens the BLOB for read or write.

Item	Description
Purpose	Opens the BLOB for read or write.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1BlobOpen(TM1P hPool, TM1V hBlob);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hBlob is the handle to the BLOB.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean.
Security	None.
Errors	TM1ErrorBlobOpenFailed
See Also	Other TM1Blob functions.

TM1BlobPut

Writes data to a BLOB.

Item	Description
Purpose	Writes data to a BLOB.
Definition	<pre>TM1IMPORT unsigned long TM1API TM1BlobPut(TM1U hUser, TM1V hBlob, TM1_INDEX x, TM1_INDEX n, CHAR *buf);</pre>
Parameters	hUser is the user handle obtained with TM1SystemOpen. hBlob is the handle to the BLOB. x is the starting location in the BLOB to write. n is the number of bytes to be written to the BLOB. buf is the pointer to memory containing data to be written.
Result	The function returns the number of bytes written successfully.
Security	None.
Errors	None.
See Also	Other TM1Blob functions.

TM1CancelClientJob

Creates a new worker thread to authenticate the thread, and cancel the process as a result of the user pressing a cancel button.

Item	Description
Purpose	Creates a new worker thread to authenticate the thread, and cancel the process as a result of the user pressing a cancel button. TM1CancelClientJob cancels, on behalf of the current client, the job of the current client operation being executed on the server.
Definition	TM1IMPORT TM1_BOOL TM1API TM1CancelClientJob(TM1U hUser, TM1V hServer)
Parameters	<i>hUser</i> is a handle to the current user. <i>hServer</i> is a handle to the current server. TM1CancelClientJob will authenticate the cancel request and cancel the current transaction.
Security	None.
Errors	None.
Result	Upon completion, a success or failure is returned by the server.
See Also	TM1UserKill

TM1ChangeSetBegin

Marks the beginning of a collection of changes.

Item	Description
Purpose	Marks the beginning of a collection of changes.
Definition	TM1V voChangeIdStr = TM1ChangeSetBegin(hPool, voServer)
Parameters	<i>hPool</i> is a pool handle obtained with TM1ValPoolCreate. <i>voServer</i>
Result	None.

Item	Description
Errors	Note that ChangeSet's cannot nest. If you call ChangeSetBegin() without having called ChangeSetEnd() from a previous Begin(), you will get an error returned. er_ChangeSetNotOpen A call was made to ChangeSetEnd() without the open ever being called. er_ChangeSetAlreadyOpen A call was made to ChangeSetOpen() without a call to close the first open change set er_ChangeSetUndoFailed A call to undo has failed. Any changes with the associated voChangeIdStr were not performed.
Example	<pre> TM1V voChangeSetId = TM1ChangeSetBegin (p.p, m_pCubeViewDlg->voServer); const char *ChangeSetId = TM1ValStringGet(m_pCubeViewDlg->hUser, voChangeSetId); // Save this value for future rollbackvoResult = TM1CubeCellSpreadViewArray(p.p,m_pCubeViewDlg->hView, voCellRange, voCellRef, voControlString); TM1ChangeSetEnd(p.p, m_pCubeViewDlg->voServer); </pre>
See Also	"TM1ChangeSetUndo" on page 61 "TM1ChangeSetEnd"

TM1ChangeSetEnd

Marks the end of a collection of changes.

Item	Description
Purpose	Marks the end of a collection of changes.
Definition	TM1ChangeSetEnd(hPool, voServer)
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. voServer
Result	This function returns a value - a TM1_INDEX which is an integer value that represents the number of changes cells since the ChangeSetBegin() call. This allows a client application to know if there is anything to "undo" in the changeset.
See Also	"TM1ChangeSetBegin" on page 59 "TM1ChangeSetUndo" on page 61

TM1ChangeSetUndo

Undoes all changes found in the log file that have the Id specified by voChangeIdStr

Item	Description
Purpose	Undoes all changes found in the log file that have the Id specified by voChangeIdStr
Definition	TM1ChangeSetUndo(hPool, voServer, voChangeIdStr)
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. voServer
Result	Returns a TM1V integer that indicates the amount of changes that were made successfully as a result of the undo function.
See Also	"TM1ChangeSetBegin" on page 59 "TM1ChangeSetEnd" on page 60

TM1ChoreExecute

Executes a TurboIntegrator chore on an IBM Cognos TM1 server.

Item	Description
Purpose	Executes a TurboIntegrator chore on an IBM Cognos TM1 server.
Definition	TM1IMPORT TM1V TM1API TM1ChoreExecute(TM1P hPool, TM1V hChore);
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hChore is a value capsule containing a valid handle to a chore defined on the TM1 server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerChores.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 0, the chore execution generated errors, otherwise the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None
Errors	None
See Also	TM1ProcessExecute

TM1ClientAdd

Adds a new client to a server.

Item	Description
Purpose	Adds a new client to a server.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientAdd(TM1P hPool, TM1V hServer, TM1V sClientName);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server to which the client will be added. sClientName is a TM1_STRING containing the name of the client to be added.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. After calling TM1ClientAdd you must assign a password to the client with the function TM1ClientPasswordAssign. After adding a client, call TM1ObjectListHandleByNameGet to get a handle to the client. It is strongly suggested that you assign a password to the client with the function TM1ClientPasswordAssign after adding a new client.
Security	The client must have ADMIN rights to the server.
Errors	<pre>TM1ErrorClientAlreadyExists TM1ErrorObjectSecurityNoAdminRights</pre>
See Also	Other TM1Client functions.

TM1ClientGroupAssign

Assigns a client to a group.

Item	Description
Purpose	Assigns a client to a group.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientGroupAssign(TM1P hPool, TM1V hClient, TM1V hGroup);</pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hClient is a handle of the client to be assigned. To retrieve the client handle from the server call TM1ObjectListHandleByNameGet. The format of this function is as follows: <pre>vResult = TM1ObjectListHandleByNameGet(hPool, TM1ServerClients(), vClientName)</pre> hGroup is a handle to the group to which the client is to be assigned. To retrieve the group handle from the server, call TM1ObjectListHandleByNameGet. The format of this function is as follows: <pre>vResult = TM1ObjectListHandleByNameGet (hPool, TM1ServerGroups(), vGroupName)</pre>
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorObjectSecurityNoAdminRights

Note: The ClientGroupAssign function rejects any attempt to place a client (user) that is a member of the SecurityAdmin group into another group.

TM1ClientGroupsAssigned

Determines whether a client is assigned to a group.

Item	Description
Purpose	Determines whether a client is assigned to a group.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientGroupIsAssigned (TM1P hPool, TM1V hClient, TM1V hGroup);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hClient is a handle to a client. hGroup is a handle to a group.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the client is assigned. If it is zero, the client is not assigned. Use the function TM1ValBoolGet to extract the Boolean.

Item	Description
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Client functions.

TM1ClientGroupRemove

Removes a client from a group.

Item	Description
Purpose	Removes a client from a group.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientGroupRemove(TM1P hPool, TM1V hClient, TM1V hGroup);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hClient is a handle of the client to be removed. hGroup is a handle to the group from which the client is to be removed.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Client functions.

TM1ClientHasHolds

Checks whether the client has hold cells or not.

Item	Description
Purpose	Checks whether the client has hold cells or not.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientHasHolds(TM1P hPool, TM1V hClient);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hClient is a client handle. Client handles are returned by the function TM1SystemServerConnect. You can also retrieve a client handle from the server list property TM1ServerClients.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the boolean is TRUE, the client has one or more hold cells.
Security	None.
Errors	None.
See Also	Other TM1ViewArray functions.

TM1ClientPasswordAssign

Assigns a new password to a client.

Item	Description
Purpose	Assigns a new password to a client.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientPasswordAssign(TM1P hPool, TM1V hClient, TM1V sPassword);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hClient is a handle to a client object. To retrieve the client handle from the server call TM1ObjectListHandleByNameGet. The format of this function is as follows: <pre>vResult = TM1ObjectListHandleByNameGet(hPool, TM1ServerClients(), vClientName);</pre> The vClientName argument should be the name you added with TM1ClientAdd. sPassword is a TM1V containing the password string. You can construct this value capsule with the functions TM1ValString or TM1ValString encrypt.
Result	Returns a TM1V containing the new password as a TM1_STRING. Use TM1ValStringGet to retrieve the string from the value capsule.
Security	Only the client whose password is being changed and clients with ADMIN privileges can assign passwords.
Errors	None.
See Also	TM1ClientAdd

TM1ConnectionCheck

Checks a connection object for consistency.

Item	Description
Purpose	Checks a connection object for consistency.
Definition	<code>TM1IMPORT TM1V TM1API TM1ConnectionCheck(TM1P hPool, TM1V hConnection);</code>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate hConnection is a handle to a connection object. This object handle is returned by TM1ConnectionCreate, or it can be retrieved from the IBM Cognos TM1 Server List object TM1ServerConnections.
Result	The function returns a Boolean 1 if the operation is successful.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1ConnectionCreate

Creates a new connection object.

Item	Description
Purpose	Creates a new connection object.
Definition	<code>TM1IMPORT TM1V TM1API TM1ConnectionCreate(TM1P hPool, TM1V hServer, TM1V sStarServerName, TM1V sUsername, TM1VsPassword);</code>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server on which the connection will be created. sStarServerName is a string value containing the name of the star server. sUsername is a string value containing the name of the IBM Cognos TM1 client. sPassword is a string value containing the password.

Item	Description
Result	This function returns a handle to a connection object. The TM1 servers on either side of the connection must be registered with the TM1 admin host that you specified when you called TM1SystemAdminHostSet. Once you have created the connection, you should populate the following connection object properties: TM1ConnectionSyncStarToPlanet - Data changed on the star server is migrated to the planet server during a synchronization. TM1ConnectionSyncPlanetToStar - Data changed on the planet server is migrated to the Star server during a synchronization.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1ConnectionDelete

Deletes a connection object.

Item	Description
Purpose	Deletes a connection object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ConnectionDelete(TM1P hPool, TM1V hConnection);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hConnection is a handle to a connection object. This object handle is returned by TM1ConnectionCreate, or it can be retrieved from the IBM Cognos TM1 Server List object TM1ServerConnections.
Result	This function deletes a connection object from the TM1 server.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1ConnectionSynchronize

Performs synchronization on a connection object.

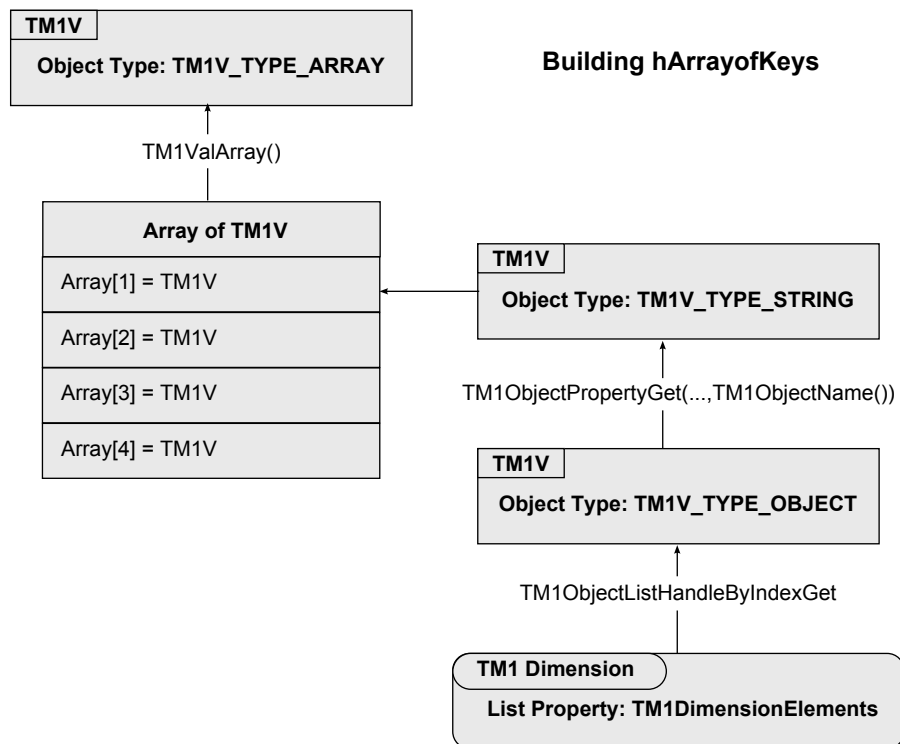
Item	Description
Purpose	Performs synchronization on a connection object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ConnectionSynchronize(TM1P hPool, TM1V hConnection);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate hConnection is a handle to a connection object. This object handle is returned by TM1ConnectionCreate, or it can be retrieved from the IBM Cognos TM1 server list property TM1ServerConnections.
Result	The function returns a Boolean 1 if the operation is successful. A successful synchronization means that cubes on both sides of the connection have the latest data.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1CubeCellDrillListGet

Returns a list of drill object process names associated with a cell.

Item	Description
Purpose	Returns a list of drill object process names associated with a cell.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellDrillListGet(TM1P hPool, TM1V hCube, TM1V hArrayOfKeys);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hCube is a handle to a cube. hArrayOfKeys is a TM1V containing an array. This is an array of element names. There should be a name of one element for each dimension in the cube. These elements, in combination, identify the exact cell whose list of drill processes you want to retrieve. The diagram below the table shows how to build this array of element names.
Result	The function returns a TM1V array which includes all the drill object process names.

Item	Description
Security	None.
Errors	TM1ErrorCubeDrillNotFound TM1ErrorCubeNumberOfKeysInvalid TM1ErrorCubeDrillInvalidStructure TM1ErrorSystemParameterTypeInvalid TM1ErrorCubeKeyInvalid
See Also	Other TM1ViewArray functions.



TM1CubeCellDrillObjectBuild

Returns a drill object associated with a cell and a drill object process name.

Item	Description
Purpose	Returns a drill object associated with a cell and a drill object process name.
Definition	<pre> TM1IMPORT TM1V TM1API TM1CubeCellDrillObjectBuild(TM1P hPool, TM1V hCube, TM1V hArrayOfKeys, TM1V sDrillProcessName); </pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hCube is a TM1V containing the handle to a cube. You can retrieve a handle to an existing cube through the IBM Cognos TM1 server list property TM1ServerCubes. hArrayOfKeys is a TM1V containing an array. This is an array of element names. There should be a name of one element for each dimension in the cube. These elements, in combination, identify the exact cell whose list of drill processes you want to retrieve. This the previous diagram that shows an array of element names.
	sDrillProcessName is a TM1V containing a string. This is the string name of the drill object process, such as }Drill_Drill Transactions. Note: The full name of the process is required, including the }Drill_ prefix.
Result	The TM1 server runs the drill object process and returns a TM1V containing an object. The object is one of the following types: TM1TypeSQLTable TM1TypeView If the returned object type is TM1TypeSQLTable, you can retrieve following properties: TM1SQLTableColumnNames TM1SQLTableColumnTypes TM1SQLTableNumberOfColumns TM1SQLTableNumberOfRows TM1SQLTableRowsetSize After using this object, you should delete it with TM1ObjectDestroy.
Security	None.
Errors	TM1ErrorObjectNotFound TM1ErrorSystemParameterTypeInvalid
See Also	TM1CubeCellDrillListGet

TM1CubeCellPickListGet

Call this function to get the PickList associated with a cell in a cube.

Item	Description
Purpose	Call this function to get the PickList associated with a cell in a cube.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellPickListGet(TM1P hPool, TM1V hCube, TM1V hArrayOfElements)</pre>
Parameters	hCube - cube which contains the cell for which you wish to retrieve a PickList hArrayOfElements - the key that describes a cell in the cube
Result	A two-element array. The first element is an enumeration indicating what type of PickList is contained in the second element. If this enumeration is 'TYPE_SUBSET' then the second element of the array is also an array with two elements. The first element is a string representing the subset's dimension's name and the second element is a string representing the subset's name. If the type enumeration is 'TYPE_STATIC' then the second element of the array is an array of strings with each string being an element of the PickList. If the type enumeration is 'TYPE_DIMENSION' then the second element of the array is a string representing a dimension name. If the type enumeration is 'TYPE_NONE' then the second element will not be present, indicating that this cell has no PickList.

TM1CubeCellsPickListGet

Call this function to get the PickList associated with cell(s) in a cube.

Item	Description
Purpose	Call this function to get the PickList associated with cell(s) in a cube.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellsPickListGet(TM1P hPool, TM1V hCube, TM1V hArrayOfCells)</pre>
Parameters	hCube - cube which contains the cell(s) for which you wish to retrieve a PickList hArrayOfCells - an array of 'hArrayOfElements' items as defined in "TM1CubeCellPickListGet" on page 70.
Result	A two element array. The first element is an array of pick list values. The second element is an array of indices into the array of pick list values for each cell in the range.

TM1CubeCellPickListExists

Call this function to check if a PickList exists or not for a given cell.

Item	Description
Purpose	Call this function to check if a PickList exists or not for a given cell.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellPickListExists(TM1P hPool, TM1V hCube, TM1V hArrayOfElements)</pre>
Parameters	hCube - cube which contains the cell that you wish to check for presence of a PickList hArrayOfElements - the key that describes a cell in the cube
Result	A Boolean indicating whether or not a PickList exists for this cell.

TM1CubeCellSpreadViewArray

Spreads data specified in sControl to a range of cells in a view.

Item	Description
Purpose	Spreads data specified in sControl to a range of cells in a view. This function uses row and column pairs to mark the starting location for the spread command.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellSpreadViewArray(TM1P hPool, TM1V hView, TM1V aCellRange, TM1V aCellRef, TM1V sControl);</pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews. aCellRange is a handle to an array value. This array contains the locations in the view to which your data will be spread. This array can contain either two values or four values. If the array contains two TM1V integer values (column1, row1), the specified cell is used as a starting point for the data spread. The remainder of the range is determined by the sControl argument. If the array contains four TM1V integer values (column1, row1, column2, row2), the paired coordinates represent the starting and ending cells of the range where the data will be spread. aCellRef is only used for Relative Proportional Spread and Relative Percent Adjustment. This TM1V contains an IBM Cognos TM1 array. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. To indicate that there is no reference cell, this parameter should be set to TM1ArrayNull() or to an array with size zero.
Parameters	<i>aCellRef</i> is the reference cell for aCellRange. Both aCellRef and aCellRange must be single cell ranges. aCellRef may refer to a cell in any cube, but the target cell must be a consolidated cell, and the consolidation must be identical to the one referenced by aCellRange. sControl is a TM1V containing a string value. This string is the spreading command. For example, S>100. For a complete list of the TM1 spreading commands, refer to “Data Spreading Syntax” on page 222.
Result	The function returns a TM1V containing three successful cases: TM1CubeCellSpreadFunctionOk() TM1CubeCellSpreadNumericCellSetOk() TM1CubeCellSpreadStringCellSetOk()
Security	None.
Errors	TM1ErrorDataSpreadFailed TM1ErrorObjectNotLoaded TM1ErrorViewNotConstructed
See Also	TM1CubeCellSpread

Item	Description
Example	<pre> void cube_spread_view_array(TM1U hUser, TM1V hServer, CHAR * sCube, CHAR * sView) { TM1P hPool; TM1V lRet = NULL; TM1V hCube = NULL; TM1V hView = NULL; TM1V hArrayColumnsNof = NULL; TM1V hArrayRowsNof = NULL; TM1V hArrayOfCells = NULL; TM1V hSpreadArray = NULL; TM1V emptyArray = TM1ArrayNull(); TM1V hLocation[2]; TM1_INDEX iRet = 0; TM1_INDEX iCtr = 0; TM1_INDEX iSize=0; TM1_INDEX iRows=0; TM1_INDEX iCols=0; TM1_INDEX iCount=0; TM1_REAL rVal = 333; int decimal; int sign; char sSpread[MAX_STRING] = "R>"; strcat(sSpread,fcvt(rVal,0, &decimal,&sign)); hPool = TM1ValPoolCreate(hUser); hCube = TM1ObjectListHandleByNameGet (hPool, hServer, TM1ServerCubes(), TM1ValString(hPool, sCube,0)); CheckError(hUser,hCube, "TM1ObjectListHandleByNameGet"); </pre>

Item	Description
Example (cont.)	<pre> hView = TM1ObjectListHandleByNameGet (hPool,hCube, TM1CubeViews(), TM1ValString (hPool, sView,0)); CheckError(hUser,hView, "TM1ObjectListHandleByNameGet"); //ViewArrayConstruct lRet = TM1ViewArrayConstruct (hPool, hView); CheckError (hUser, lRet, "TM1ViewArrayConstruct"); //Get number of columns on view hArrayColumnsNof = TM1ViewArrayColumnsNof (hPool, hView); CheckError (hUser, hArrayColumnsNof, "TM1ViewArrayColumnsNof"); //Get number of rows on view hArrayRowsNof = TM1ViewArrayRowsNof (hPool, hView); CheckError (hUser, hArrayRowsNof, "TM1ViewArrayRowsNof"); //Get View Title Elements lRet = TM1ObjectPropertyGet (hPool, hView, TM1ViewTitleElements()); CheckError (hUser, lRet, "TM1ObjectPropertyGet- TM1ViewTitleElements"); //Set location for spread hLocation[0] = TM1ValIndex (hPool, 2); //column hLocation[1] = TM1ValIndex (hPool,4); //row hArrayOfCells = TM1ValArray (hPool, hLocation, 2); </pre>

Item	Description
Example (cont.)	<pre> //Clear cells to zero lRet = TM1CubeCellSpreadViewArray(hPool, hView, hArrayOfCells, TM1ArrayNull(), TM1ValString(hPool, "C>",0)); CheckError(hUser,lRet, "TM1CubeCellSpreadViewArray"); //Spread value lRet = TM1CubeCellSpreadViewArray (hPool, hView, hArrayOfCells, TM1ArrayNull(),TM1ValString (hPool, sSpread,0)); CheckError(hUser,lRet, "TM1CubeCellSpreadViewArray"); for (iCount = 1; iCount<=iCols; iCount++){ //Get View Array Value lRet = TM1ViewArrayValueGet (hPool, hView, TM1ValIndex (hPool, iCount), TM1ValIndex (hPool, 4)); //CheckError (hUser, lRet, "TM1ViewArrayValueGet"); //This should always be the case for this view but check anyway if (TM1ValType (hUser, lRet) == TM1ValTypeReal()){ if(TM1ValRealGet(hUser,lRet) != rVal){ cout << "Spread Value: " << rVal << endl; CheckError(hUser,TM1ValBool(hPool,FALSE), "TM1CubeCellSpreadViewArray"); } //end if } //end if } //end second for //Destroy Array lRet = TM1ViewArrayDestroy (hPool, hView); CheckError (hUser, lRet, "TM1ViewArrayDestroy"); TM1ValPoolDestroy(hPool); } //end cube_spread_view_array </pre>
Example (cont.)	<pre> } //end if } //end second for //Destroy Array lRet = TM1ViewArrayDestroy (hPool, hView); CheckError (hUser, lRet, "TM1ViewArrayDestroy"); TM1ValPoolDestroy(hPool); } //end cube_spread_view_array </pre>

TM1CubeCellSpread

Spreads data to an array of cells in one or more cubes.

Item	Description
Purpose	Spreads data to an array of cells in one or more cubes. This function uses cube handles and element handles to mark the starting location for the spread command. No view handle is required for this function.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellSpread(TM1P hPool, TM1V hServer, TM1V vArrayOfCells, TM1V vCellReference, TM1V sSpreadData);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect. vArrayOfCells is a TM1V containing an array of cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1, Array2, Array3..., Arrayn};</pre> Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle}; Array2 = {CubeHandle2, ElementHandle, ElementHandle..., ElementHandle}; The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the IBM Cognos TM1 dimension list property TM1DimensionElements.
Parameters	<i>vCellReference</i> is the reference cell for vArrayOfCells. Both vCellReference and vArrayOfCells must be single cell ranges. vCellReference may refer to a cell in any cube, but the target cell must be a consolidated cell, and the consolidation must be identical to the one referenced by vArrayOfCells. vCellReference is only used for Relative Proportional Spread and Relative Percent Adjustment. It is ignored in any other case. vCellReference is a TM1V containing a TM1 array. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. It has the general form [cubehandle, elementhandle1, elementhandle2, elementhandle3...]. To indicate that there is no reference cell, this parameter should be set to TM1ArrayNull or to an array with size zero. sSpreadData is a TM1V containing a string value. This string is the spreading command. For example, S>100. For a complete list of the IBM Cognos TM1 spreading commands, refer to "Data Spreading Syntax" on page 222.

Item	Description
Result	Use this function to spread a value when the client application does not have a view handle available. For example, if you are spreading values in a spreadsheet that contains DBRW functions, you should use this function. This function ignores the direction codes in the control string. It is incumbent on the programmer to build the vArrayOfCells array with the correct cell range. The function returns a TM1V containing three successful cases: • TM1CubeCellSpreadFunctionOk() indicates the spread was performed successfully. • TM1CubeCellSpreadNumericCellSetOk() indicates that the control string was a number and that it was successfully entered into the numeric cell. • TM1CubeCellSpreadStringSetOk() indicates that the string was successfully entered into the string cell.
Security	None.
Errors	TM1ErrorDataSpreadFailed() TM1ErrorCubeCellWriteStatusCubeNoWriteAccess() TM1ErrorCubeCellWriteStatusCubeReserved() TM1ErrorCubeCellWriteStatusCubeLocked() TM1ErrorCubeCellWriteStatusRuleApplies() TM1ErrorCubeCellWriteStatusElement IsConsolidated() TM1ErrorCubeCellWriteStatusElement NoWriteAccess() TM1ErrorCubeCellWriteStatusElementReserved() TM1ErrorCubeCellWriteStatusElementLocked()
See Also	TM1CubeCellSpreadViewArray

Item	Description
Example	<pre> void cube_spread_cell_array(TM1U hUser, TM1V hServer, CHAR * sCube) { // Spreading example using TM1CubeCellSpread TM1P hPool; TM1VlReturn = NULL; TM1V lRet = NULL; TM1V hCube = NULL; TM1V hDim = NULL; TM1V hElm[TM1_MAXIMUM_DIMENSIONS]; TM1V hElmArray[1]; TM1V hElm_han = NULL; TM1V hArrayOfCells = NULL; TM1_INDEX iCtr = 0; TM1_INDEX NumOfDims = 0; TM1_REAL rVal = 2321; int decimal; int sign; char sSpread[MAX_STRING] = "R"; strcat(sSpread,fcvt(rVal,0,&decimal,&sign)); // Create Pool hPool = TM1ValPoolCreate(hUser); // Get Cube Handle hCube = TM1ObjectListHandleByNameGet (hPool, hServer,TM1ServerCubes(), TM1ValString(hPool,sCube,0)); CheckError(hUser,hCube, "TM1ObjectListHandleByNameGet- TM1ServerCubes"); </pre>

Item	Description
Example (cont.)	<pre> // Get number of dimensions lRet = TM1ObjectListCountGet(hPool,hCube, TM1CubeDimensions()); NumOfDims = TM1ValIndexGet(hUser,lRet); // Make array for getting cell value hElm_han = TM1ValArray(hPool, hElm,NumOfDims); // Loop through and get element handles for (iCtr=1; iCtr<=NumOfDims; iCtr++) { // Get Dimension handles hDim = TM1ObjectListHandleByIndexGet (hPool,hCube,TM1CubeDimensions(), TM1ValIndex(hPool,iCtr)); CheckError(hUser,hDim, "TM1ObjectListHandleByIndexGet- TM1CubeDimensions"); // Get Element Handles hElm[iCtr] = TM1ObjectListHandleByIndexGet (hPool,hDim,TM1DimensionElements(), TM1ValIndex(hPool,(2))); CheckError(hUser,hElm[iCtr], "TM1ObjectListHandleByIndexGet- TM1DimensionElements"); // Set array for getting cell value TM1ValArraySet(hElm_han, hElm[iCtr],iCtr); } // End For Loop // put cube handle in first spot of elm arrays for spread hElm[0] = hCube; // Initailize TM1ValArray at first level of hElmArray[0] hElmArray[0] = TM1ValArray(hPool,hElm,NumOfDims + 1); CheckError(hUser,hElmArray[0],"TM1ValArray"); </pre>

Item	Description
Example (cont.)	<pre> // set TM1ValArray into TM1ValArray hArrayOfCells = TM1ValArray(hPool,hElmArray,1); // clear values lRet = TM1CubeCellSpread(hPool, hServer, hArrayOfCells,TM1ArrayNull(), TM1ValString(hPool,"C",0)); CheckError(hUser,lRet,"TM1CubeCellSpread"); // Execute the spread lRet = TM1CubeCellSpread(hPool, hServer, hArrayOfCells,TM1ArrayNull(), TM1ValString(hPool, (CHAR*)(sSpread),0)); CheckError(hUser,lRet,"TM1CubeCellSpread"); // Confirm value lRet = TM1CubeCellValueGet(hPool, hCube, hElm_han); if(TM1ValRealGet(hUser, lRet) != rVal){ // Fail CheckError(hUser,TM1ValBool(hPool,FALSE), "TM1CubeCellSpread"); } TM1ValPoolDestroy(hPool); } //end cube_spread_cell_array </pre>

TM1CubeCellSpreadStatusGet

Checks the status of the cells of an IBM Cognos TM1 view or a TM1 cube.

Item	Description
Purpose	Checks the status of the cells of an IBM Cognos TM1 view or a TM1 cube.
Definition	<pre> TM1IMPORT TM1V TM1API TM1CubeCellsSpreadStatusGet(TM1P hPool, TM1V hServer, TM1V hCells, TM1V hCellRange); </pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle to a TM1 server object. This handle is returned by the function TM1SystemServerConnect. hCells is a TM1V containing one of two values: • A two dimensional array of cell references of the form: • [[cubehandle1, elemhandle, elemhandle,...], • [cubehandle2, elemhandle, elemhandle,...]] The cubehandles can refer to different cubes. In this case, the status of cells within multiple cubes is returned, and the hCellRange parameter is ignored. • A handle to a view, which will be used to extract the range of cells, defined by hCellRange. The function returns the status of all the cells in this range.
Parameters	hCellRange is a TM1V containing one of the following values: • NULL Object - If hCells is an array, hCellRange should be set to a NULL object. • A TM1V containing an array. This argument is used only when hCells is a handle to a view. This is an array of indices indicating the upper left and lower right cells coordinates of a range within the view. It has the general form [column1, row1, column2, row2]. If column2, row2 are not specified then the function returns the status of the cell defined by [column1, row1] inside the view. • TM1ArrayNull() or an array with zero elements then the function returns the status of all the cells in the view.
Result	The function returns a TM1V containing an array of indices. There is one element in the array for each cell specified in the hCells and hCellRange arguments. If hCells is an array of cells then the items in the returned array will match the ones in hCells. If hCells is a view handle then the items in the array correspond to the cells in the view range as shown in the figure following the table.
Result (cont.)	Each value in the returned array is one of the following: • TM1CubeCellSpreadStatusHeld indicates the cell is being held and will be ignored when included in all the spreading operations except RELEASE and RELEASE ALL. • TM1CubeCellSpreadStatusHeldConsolidation indicates the cell's value will not be affected when this cell is included in a spreading function. (Consolidated values are not directly changed by spreading data. They may be recalculated if their component leaf cells are modified by the spreading function.) • TM1CubeCellSpreadStatusWritable indicates the cell's value will be affected when this cell is included in a spreading function.
Security	None.

Item	Description
Errors	TM1ErrorSystemValueInvalid TM1ErrorObjectNotLoaded TM1ErrorViewNotConstructed TM1ErrorSystemValueInvalid TM1ErrorSystemParameterTypeInvalid
See Also	TM1CubeCellSpreadViewArray TM1CubeCellSpread

		Region					
Account	Month	Denmark	Norway	Sweden	Belgium	Netherlands	Germany
Sales	1Quarter	4000	600	900	600	7000	6000
	Jan	900	600 (0)	700 (1)	300 (2)	888 (3)	890
	Feb	7900	8890 (4)	9000 (5)	7700 (6)	700 (7)	789
	March	677	6886 (8)	3243 (9)	67 (10)	7676 (11)	8000

TM1CubeCellValueGet

Retrieves the value of a cell from a cube.

Item	Description
Purpose	Retrieves the value of a cell from a cube.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCellValueGet(TM1P hPool, TM1V hCube, TM1V hArrayOfElements);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCube is a handle of the cube from which the data will be retrieved. hArrayOfElements is an array of element handles, one from each of the dimensions of the cube, in the same order as the dimensions themselves.
Result	Returns the value stored in the cell specified.
Security	The client must have at least READ access to the cube, and to all the elements that identify the cell.
Errors	TM1ErrorCubeKeyInvalid TM1ErrorCubeNumberOfKeysInvalid TM1ErrorObjectSecurityNoReadRights

Item	Description
See Also	TM1CubeCellValueSet

TM1CubeCellsValueGet

Retrieves the value of an array of cells from a cube.

Item	Description
Purpose	Retrieves the value of an array of cells from a cube.
Definition	TM1V vValues = TM1CubeCellsValueGet(TM1P hPool, TM1V hCube, TM1V vArrayOfCells)
Parameters	hPool:a pool handle hView:a cube handle vArrayOfCells:array of cells, each cell is an array of coordinate of element handles in the exact same order as the dimension in the cube.
Result	An array of cell values, each array contains an array of cell values, has spread hold status, etc.
Example	Array(0)// cell 1 Array(0)element 1 handle Array(1)element 2 handle ... Array(n-1)element n handle Array(1)// cell 2 ... Array(n-1)// n number of cells

TM1CubeCellValueSet

Updates the value of a cell in a cube.

Item	Description
Purpose	Updates the value of a cell in a cube.
Definition	TM1IMPORT TM1V TM1API TM1CubeCellValueSet(TM1P hPool, TM1V hCube, TM1V hArrayOfElements, TM1V hValue);

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCube is a handle of the cube from which the data will be retrieved. hArrayOfElements is an array of element handles, one from each of the dimensions of the cube, in the same dimension order as that with which the cube is defined. hValue is the value to be stored in the cell.
Result	Returns the new value of the cell. The TM1V may contain a TM1_STRING, a TM1_BOOL, a TM1_REAL or a TM1_ERROR. Be sure to check the type before using a return value from this function. A TM1_BOOL whose value is 0, or a TM1_ERROR indicates a failure.
Security	The client must have at least WRITE access to the cube and to all the dimensions that identify the cell.
Errors	TM1ErrorCubeKeyInvalid TM1ErrorCubeNumberOfKeysInvalid TM1ErrorCubeCellValueTypeMismatch TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectSecurityNoWriteRights
See Also	TM1CubeCellValueGet

TM1CubeCellsValueSet

Updates the value of an array of cells in a cube.

Item	Description
Purpose	Updates the value of an array of cells in a cube.
Definition	TM1V vRet = TM1CubeCellsValueSet(TM1P hPool, TM1V hCube, TM1V vArrayOfCells, TM1V vValues)
Parameters	hPool:a pool handle hCube:a cube handle vArrayOfCells:array of cells, each cell is an array of coordinate of element handles in the exact same order as the dimension in the cube. vValues:an array of values for the corresponding cells to be written back to the cube cells
Result	vRet: an array of status for each cell writeback. For those cells that writeback failed, the value would be TM1ValError in the corresponding array location.

TM1CubeCreate

Creates a new cube.

Item	Description
Purpose	Creates a new cube.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubeCreate(TM1P hPool, TM1V hServer, TM1V hArrayOfDimensions);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle to the server on which to create the cube. hArrayOfDimensions is an array of between 2 and 16 dimension handles with which to construct the cube. Specify free dimensions by setting the corresponding element handle to zero.
Result	The function returns a handle to the newly created cube. The cube must still be registered before other applications can access it.
Security	The client must have at least READ access to the dimensions used to create the cube.
Errors	<pre>TM1ErrorCubeDimensionInvalid TM1ErrorCubeNotEnoughDimensions TM1ErrorCubeTooManyDimensions TM1ErrorObjectSecurity NoReadRights TM1ErrorCubeCreationFailed</pre>
See Also	TM1DimensionCreateEmpty

TM1CubeDimensionListGet

Returns a list of dimensions in a cube in order.

Item	Description
Purpose	Returns a list of dimensions in a cube in order.
Definition	<pre>TM1V vDimensionList = TM1CubeDimensionListGet(TM1V hCube)</pre>
Parameters	hServer is a cube handle

Item	Description
Result	vDimensionList is an array of dimensions of the specified cube in exact order they exist in the cube, each array include dimension handle, name, element count, public subset count, private subset count, etc. Note the(index + 1) of top level array should match the index of dimension in the cube (1-based index in IBM Cognos TM1 server). Array ();// array (index+1)matching dimension index in the cube Array(CDL_DIMENSIONHANDLE): Array(CDL_DIMENSIONNAME): Array(CDL_ELEMENTCOUNT): Array(CDL_LASTUPDATETIME): Array(CDL_NUMBEROFLEVELS):

TM1CubeListByNamesGet

Returns a list of cubes with names, etc.

Item	Description
Purpose	Returns a list of cubes with names, etc.
Definition	hServer, TM1V vCubeNames
Parameters	hServer is a server handle vCubeNames is an array of cube names.
Result	vCubeList is an Array of cubes in the server, each array includes cube handle, name, number of dimensions, etc. Array (): Array(CL_CUBEHANDLE): Array(CL_CUBENAME): Array(CL_NUMBEROFDIMENSIONS): Array(CL_ISCUBEVIRTUAL): Array(CL_LASTUPDATETIME): ...

TM1CubeListGet

Returns a list of cubes.

Item	Description
Purpose	Returns a list of cubes.
Definition	TM1V hServer, TM1V iFlag
Parameters	hServer: a server handle iFlat: control cubes/no control cubes/both (CL_GET_CONTROLCUBES CL_GET_NONCONTROLCUBES)
Result	vCubeList: an Array of cubes in the server, each array includes cube handle, name, number of dimensions, etc. Array (): Array(CL_CUBEHANDLE): Array(CL_CUBENAME): Array(CL_NUMBEROFDIMENSIONS): Array(CL_ISCUBEVIRTUAL): Array(CL_LASTUPDATETIME): ...

TM1CubePerspectiveCreate

Calculates a perspective of a cube.

Item	Description
Purpose	Calculates a perspective of a cube. A perspective can be thought of as a sub-cube of a cube. It is defined by choosing one or more free dimensions, which will be the dimensions of the resulting sub-cube. The rest of the dimensions are fixed by choosing a specific element from each.
Definition	TM1IMPORT TM1V TM1API TM1CubePerspectiveCreate(TM1P hPool, TM1V hCube, TM1V hArrayOfElementTitles);

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCube is a handle of the Cube from which the perspective will be built. hArrayOfElementTitles is an array of element handles, one from each of the dimensions of the cube, in the dimension order of the cube. Specify free dimensions by setting the corresponding element handle to zero. See example for TM1CubeCreate for outline to create hArrayOfElementTitles.
Result	The function returns a TM1_OBJECT handle to the perspective generated. The perspective created is stored with the cube. Any reference to a cell in the perspective will be satisfied from the perspective.
Security	The client must have at least READ access to the cube, and to all the fixed elements.
Errors	TM1ErrorCubeKeyInvalid TM1ErrorCubeNumberOfKeysInvalid TM1ErrorCubePerspectiveAllSimpleElements TM1ErrorObjectSecurityNoReadRights TM1ErrorCubePerspectiveCreationFailed
See Also	See the following properties: TM1CubePerspectivesMaxMemory TM1CubePerspectivesMinTime
Example	<pre> maxDim = TM1ValIndexGet (hUser, vIndex); //initialize hArrayOfElements for (e=1; e<maxDim+1; e++){ dimArray[e] = TM1ObjectNull();} //get Array of Elements for (e=1; e<maxDim+1; e++){ hDimension = TM1ObjectListHandleByIndexGet (hPool, hCubeCopy, TM1CubeDimensions(), TM1ValIndex (hPool, e)); hElement = TM1ObjectPropertyGet (hPool, hDimension, TM1DimensionTopElement()); TM1ValArraySet (hArrayOfElements, hElement, e);} nReturnCode = T M1CubePerspectiveCreate (hPool, hCubeCopy, hArrayOfElements()); </pre>

TM1CubePerspectiveDestroy

Deletes a perspective of a cube.

Item	Description
Purpose	Deletes a perspective of a cube. A perspective can be thought of as a sub-cube of a cube. It is defined by choosing one or more free dimensions, which will be the dimensions of the resulting sub-cube. The rest of the dimensions are fixed by choosing a specific element from each.
Definition	<pre>TM1IMPORT TM1V TM1API TM1CubePerspectiveDestroy(TM1P hPool, TM1V hCube, TM1V hArrayOfElements);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hCube is a handle of the cube from which the perspective will be deleted. hArrayOfElements is an array of element handles, one from each of the dimensions of the cube, in the dimension order of the cube. Specify free dimensions by setting the corresponding element handle to zero.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. This function deletes the perspective.
Security	The client must have at least READ access to the cube.
Errors	TM1ErrorCubeNumberOfKeysInvalid
See Also	TM1CubePerspectiveCreate

TM1CubeShowsNulls

Returns whether the cube has the UNDEFVALS rule.

Item	Description
Purpose	Returns whether the cube has the UNDEFVALS rule.
Definition	<pre>TM1CubeShowsNulls(TM1P hPool, TM1V hCube);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCube is a Cube.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the cube has the UNDEFVALS rule defined. The default behavior of IBM Cognos TM1 cubes is to treat zeros as equivalent to nulls: zeros are not stored in the cube, and empty locations are displayed as zero. The UNDEFVALS rule, if present on the cube, causes the the cube to distinguish zeros and nulls, treating zeros as regular numeric data. An UNDEFVALS cube will store zero values, and will display blanks for empty locations.
Errors	None Except for the ParameterTypeInvalid error that results if any of the object APIs are called with the wrong object type
See Also	TM1CubeCellValueUndefined

TM1CubeTimeLastInvalidated

Cube Property

Item	Description
Purpose	Cube Property
Definition	TM1IMPORT TM1V TM1API TM1CubeTimeLastInvalidated(void);
Parameters	None
Result	The value of this property is a timestamp indicating the most recent cache invalidation for the cube. Cache invalidation occurs when events like dimension update, rule recompilation, data write, etc. require that cached information be discarded. The format of the timestamp is YYYYMMDDHHmmSS.
Security	Any
Errors	None

TM1DataReservationAcquire

Requests a DR for a specific IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) cube, user and tuple.

If there is an existing reservation owned by a different user whose region overlaps the requested reservation, then the reservation request will be rejected unless the bForce flag is used. If the bForce flag is true and the user running the API has the

DataReservationOverride capability, then any conflicting reservations will be released and the new reservation will be granted.

Syntax

```
TM1DataReservationAcquire(TM1P hPool, TM1V hCube, TM1VhClient, TM1V bForce,  
TM1V elementArray);
```

Parameter	Description
<i>hPool</i>	Standard memory pool used by all API commands.
<i>hCube</i>	Handle to the cube you want to access.
<i>hClient</i>	The owner to use for the reservation
<i>bForce</i>	Boolean value that determines the behavior if the requested reservation conflicts with an existing reservation. If set to 0 (false), then the request is rejected if it conflicts with an existing reservation. If set to 1 (true), then the function replaces any conflicting reservations.
<i>elementArray</i>	Array of element handles that define the tuple, the order must match the dimension order.

Return Value

Boolean value of true if the request was granted or false otherwise.

Possible Errors

- TM1ErrorCubeNumberOfKeysInvalid
- TM1ErrorObjectHandleInvalid
- TM1ErrorCubeKeyInvalid
- TM1ErrorObjectSecurityNoReserveRights

TM1DataReservationGetAll

Determines which Data Reservations are currently held on a IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) cube.

The client parameter is optional. If it is not supplied (the parameter is set to TM1ObjectNull), then all the DRs on the cube are returned.

If the client parameter is supplied, then only the DRs held by that particular user are returned.

Syntax

`TM1DataReservationGetAll(TM1P hPool, TM1V hCube, TM1VhClient);`

Parameter	Description
<code>hPool</code>	Standard memory pool used by all API commands.
<code>hCube</code>	Handle to the cube you want to access.
<code>hClient</code>	Optional handle for the user you want to query for.

Return Value

Array of DR data with the following format:

- [1] Cube name (TM1ValTypeString)
- [2-n] Array of DR information (TM1ValTypeArray)
 - [1] Creation Time
 - [2] User name (TM1ValTypeString)
 - [3-n] Array of element names defining the tuple (TM1ValTypeArray)
 - [1-n] Element name (TM1ValTypeString)

Possible Errors

`TM1ErrorObjectHandleInvalid`

TM1DataReservationGetConflicts

Determines which reservations currently held on a IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) cube will conflict with the specified client (user) and address.

This command can be used to gather the information needed to determine why an attempt to acquire a reservation failed, assuming the reservation that caused the denial is still there.

Syntax

`TM1DataReservationGetConflicts(TM1P hPool, TM1V hCube, TM1V hClient, TM1V elementArray)`

Parameter	Description
<code>hPool</code>	Standard memory pool used by all API commands
<code>hCube</code>	Handle to the cube you want to access
<code>hClient</code>	The client (user) to compare against current reservation owners.
<code>elementArray</code>	Array of element handles that define the tuple to compare against. The order must match the dimension order

Return Value

Returns an array of DR data with the following format:

- [1] Cube name (TM1ValTypeString)
- [2-n] Array of DR information (TM1ValTypeArray)
 - [1] Creation Time
 - [2] User name (TM1ValTypeString)
 - [3-n] Array of element names defining the tuple (TM1ValTypeArray)
 - [1-n] Element name (TM1ValTypeString)

TM1DataReservationRelease

Releases an existing DR for a specific IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) cube, user and tuple.

The owner used for hClient must match the holder of the DR for the command to succeed unless the user invoking the API has the DataReservationOverride capability enabled.

The addresses supplied must be an exact match.

Syntax

```
TM1DataReservationRelease(TM1P hPool, TM1V hCube, TM1V hClient, TM1V  
elementArray);
```

Parameter	Description
hPool	Standard memory pool used by all API commands.
hCube	Handle to the cube you want to access.
hClient	The owner of the reservation.
elementArray	Array of element handles that define the tuple. The order must match the dimension order.

Return Value

Boolean value of true if the request succeeded or false otherwise. Not finding the reservation is a failure and returns false. Insufficient privilege is handled as an error.

Possible Errors

- TM1ErrorCubeNumberOfKeysInvalid
- TM1ErrorObjectHandleInvalid
- TM1ErrorCubeKeyInvalid
- TM1ErrorObjectSecurityNoAdminRights
- TM1ErrorObjectSecurityNoReserveRights

TM1DataReservationReleaseAll

Releases multiple Data Reservations for the specified IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) user.

The specified address tuple specifies the starting point for the search. All reservations owned by the specified user fully contained within the region defined by the address are released. Any reservation that overlaps the address but is not fully contained is not released.

Specifying a NULL client will remove reservations for all users. If the owner is not the same as the user executing the command, then the user must have the DataReservationOverride capability. Attempts to execute this command for a different user or all users without the override capability will be rejected without searching for existing reservations.

An administrator can release all reservations on a cube by specifying a NULL client and wildcards for every element in the address.

Syntax

```
TM1DataReservationReleaseAll(TM1P hPool, TM1V hCube, TM1V hClient, TM1V  
elementArray);
```

Parameter	Description
hPool	Standard memory pool used by all API commands.
hCube	Handle to the cube you want to access.
hClient	The owner of the reservation.
elementArray	Array of element handles that define the starting point for the release operation. The order must match the dimension order.

Return Value

Boolean value of true if there were no errors.

Possible Errors

- TM1ErrorCubeNumberOfKeysInvalid
- TM1ErrorObjectHandleInvalid
- TM1ErrorCubeKeyInvalid
- TM1ErrorObjectSecurityNoReserveRights
- TM1ErrorObjectSecurityNoAdminRights

TM1DataReservationValidate

Validates all the Data Reservations on a IBM Cognos TM1 IBM Cognos Analytic Server (ICAS) cube.

Any reservation owned by a client (user) that no longer exists will be removed.

Syntax

```
TM1DataReservationValidate(TM1P hPool, TM1V hCube);
```

Parameter	Description
hPool	Standard memory pool used by all API commands
hCube	Handle to the cube we want to access

Return Value

Boolean value of true.

Possible Errors

TM1ErrorObjectNotFound (invalid cube)

TM1DimensionAttributesGet

Returns a list of a dimension's attributes.

Item	Description
Purpose	Returns a list of a dimension's attributes.
Definition	TM1V vAttributes = TM1DimensionAttributesGet (TM1VhDim)
Parameters	hDim:a handle to a dimension object.
Result	vAttributes:an array of list of attributes, each attribute is an array containing handle, type and name. Array() Array(DA_HANDLE) Array(DA_TYPE) Array(DA_NAME)
See Also	All TM1Dimension functions.

TM1DimensionCreateEmpty

Creates an empty dimension.

Item	Description
Purpose	Creates an empty dimension.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionCreateEmpty(TM1P hPool, TM1V hServer);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server in which to create the dimension.
Result	The function returns a handle to the empty dimension object. When you create a new dimension, this is the first function you call. The complete sequence for creating a registered dimension is as follows: 1. Call TM1DimensionCreateEmpty. This function returns a handle to an empty dimension. 2. Populate the dimension with simple elements by calling TM1DimensionElementInsert. You add consolidated elements by calling TM1DimensionElementComponentAdd. 3. Once the dimension has been populated, call TM1DimensionCheck to verify the integrity of the new dimension. 4. If the integrity is intact, register the dimension with TM1ObjectRegister.
See Also	All TM1Dimension functions.

TM1DimensionCheck

Checks a dimension for consistency.

Item	Description
Purpose	Checks a dimension for consistency.
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionCheck(TM1P hPool, TM1V hDimension);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hDimension is a handle to the dimension to be checked.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the dimension has consistency and can be registered on the server. Use the function TM1ValBoolGet to extract the Boolean. This function cannot be used with registered dimensions.
Errors	<pre>TM1ErrorDimensionHasCircularReferences TM1ErrorDimensionHasNoElements TM1ObjectIsRegistered</pre>
See Also	All TM1Dimension functions.

TM1DimensionElementComponentAdd

Adds a component to a consolidated element.

Item	Description
Purpose	Adds a component to a consolidated element.
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionElementComponentAdd(TM1P hPool, TM1V hElement, TM1V hComponent, TM1V rWeight);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hElement is a handle to the consolidated element to which the component will be added. hComponent is a handle to the element to be added as a component. rWeight is a real value containing the weight of the component. The default is 1.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. The new component is inserted in the dimension. This function cannot be used with registered dimensions. Note: : If you try to put a Consolidated element under an existing N-level element, the N-level element changes to a Consolidated element and any data in the original N-level element will be lost.
Security	Because the dimension being changed is unregistered, no security considerations apply.
Errors	<pre>TM1ErrorDimensionElementComponent AlreadyExists TM1ErrorDimensionElementComponent NotNumeric TM1ErrorDimensionCircularReferences TM1ErrorDimensionElement NotConsolidated TM1ErrorObjectIsRegistered</pre>
See Also	All TM1Dimension functions.

Item	Description
Example	To add a component to a consolidated element, follow these steps: 1. Create a simple element with TM1DimensionElementInsert 2. Create a consolidated element with TM1DimensionElementInsert 3. Call TM1DimensionElementComponentAdd <pre> TM1V hConsolidatedElement, hSimpleElement; TM1V hDimension, hElementAfter; hDimension = TM1ObjectListHandleByNameGet (hPool, hServer, TM1ServerDimensions(), TM1ValString (hPool, "nameofDim",0)); hElementAfter = TM1ObjectNull(); hConsolidatedElement = TM1DimensionElementInsert (hPool, hDimension, hElementAfter, TM1ValString (hPool, name, 0), TM1TypeElementConsolidated()); hSimpleElement = TM1DimensionElementInsert (hPool, hDimension, hElementAfter, TM1ValString (hPool, name, 0), TM1TypeElementSimple()); nReturnHandle = TM1DimensionElementComponentAdd (hPool, hConsolidatedElement, hSimpleElement, TM1ValReal (hPool,rWeight)); </pre>

TM1DimensionElementComponentDelete

Deletes a component of a consolidated element.

Item	Description
Purpose	Deletes a component of a consolidated element.
Definition	<pre> TM1IMPORT TM1V TM1API TM1DimensionElementComponentDelete(TM1P hPool, TM1V hCElement, TM1V hElement); </pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCElement is a handle to the consolidated element from which the component will be deleted. hElement is a handle to the element to delete from the consolidated element.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. The component is deleted from the dimension. This function cannot be used with registered dimensions.
Security	Must have at least write rights.
Errors	TM1ErrorDimensionElement ComponentDoesNotExist TM1ErrorDimensionElement NotConsolidated TM1ErrorObjectIsRegistered TM1ObjectSecurityNoWriteRights
See Also	All TM1Dimension functions.

TM1DimensionElementComponentWeightGet

Item	Description
Purpose	Retrieves the weight of a component of a consolidated element.
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionElementComponentWeightGet (TM1PhPool, TM1V hCElement, TM1V hElement);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCElement is a handle to a consolidated element. hElement is a handle to the component within the consolidated element whose weight is sought.
Result	The function returns a real value. This value is the weight of the component in the consolidation. The default weight of a component is 1.
Errors	TM1ErrorDimensionElementComponent DoesNotExist TM1ErrorDimensionElementNotConsolidated
See Also	Other TM1Dimension functions.

TM1DimensionElementDelete

Item	Description
Purpose	Deletes an element from a dimension.
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionElementDelete(TM1P hPool, TM1V hElement);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hElement a handle to an element in the dimension.
Result	The function deletes all instances of the element from the dimension. For example, if the element appears in two different consolidations in the same dimension, both instances are deleted. This function can only be performed on unregistered dimensions. To delete an element from an existing dimension, follow these steps: 1. Get the handle to the dimension you want to update. Typically, you would use a TM1ObjectListHandle call to do this. 2. Make a copy of the dimension with TM1ObjectDuplicate. 3. Delete the unwanted element from the copy with TM1DimensionElementDelete. 4. Call TM1DimensionUpdate to replace the old dimension with the new one.
Security	At least have WRITE rights.
Errors	<pre>TM1ErrorObjectIsRegistered TM1ErrorDimensionElementDoesNotExist TM1ErrorObjectSecurityNoWriteRights</pre>
See Also	TM1DimensionElementInsert

TM1DimensionElementInsert

Item	Description
Purpose	Inserts an element in a dimension.
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionElementInsert(TM1P hPool, TM1V hDimension, TM1V hElementAfter, TM1V sName, TM1V vType);</pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hDimension is a handle of the dimension into which the element is inserted. hElementAfter is a handle to the element after which the new element is inserted. If the handle is TM1ObjectNull(), the new element is inserted at the beginning of the dimension. _ sName is a string that specifies the name of the element. vType is an integer specifying the type of element. It can be: TM1TypeElementSimple() TM1TypeElementConsolidated() TM1TypeElementString() TM1TypeElement() will bring up an index which can be compared against the above types of elements.
Result	The function returns a handle to the inserted element if the operation is successful. The new component is inserted in the dimension. This function cannot be used with registered dimensions. To update registered dimensions, follow these steps: Create a null handle. temp = TM1ObjectNull(); Make an unregistered copy. hDupDim = TM1ObjectDuplicate (hPool, hDimension); Call the function. hElement = TM1DimensionElementInsert (hPool, hDupDim, temp, TM1ValString (hPool, "string", 0), TM1TypeElementSimple()); where Simple is Consolidated, String or Simple Call TM1DimensionUpdate to overwrite the registered dimension with the newly unregistered dimension. TM1DimensionUpdate (hPool, hDimension, hDupDim); Note: Depending on the order in which you define the elements, they expand slightly differently. See the example below for details. Note: If you try to put a Consolidated element under an existing N-level element, the N-level element changes to a Consolidated element and any data in the original N-level element will be lost.
Security	Since the dimension being changed is unregistered, no security considerations apply.
Errors	TM1ErrorDimensionElementAlreadyExists TM1ErrorObjectIsRegistered

Item	Description
See Also	All TM1Dimension functions.
Example	In order: The order in which you define elements and components affects how the system displays the result. For example, you can define the elements in this order: hElement cotton candy has the child sour cotton candy.hElement sour cotton candy has children green apple and lemon.hElement cotton candy has the child sweet cotton candy.hElement sweet cotton candy has children grape and cherry.
	<pre> TM1P hPool; TM1V hDimension, hElementAfter, hElement, hComponent, hReturn; //get your own user and server handles hPool = TM1ValPoolCreate (hUser); hDimension = TM1DimensionCreateEmpty (hPool, hServer); hElementAfter = TM1ObjectNull (); hElement = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "cotton candy",0), TM1TypeElementConsolidated()); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "sour cotton candy",0), TM1TypeElementConsolidated()); </pre>

Item	Description
Example (cont.)	<pre> hReturn = TM1DimensionElementComponentAdd (hPool,hElement, hComponent, 1); hElement = TM1ObjectListHandleByNameGet (hPool, hDimension, TM1DimensionElements(), TM1ValString (hPool, "sour cotton candy", 0)); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "green apple",0), TM1TypeElementSimple()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "lemon",0), TM1TypeElementSimple()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); </pre>
	<pre> hElement = TM1ObjectListHandleByNameGet (hPool, hDimension, TM1DimensionElements(), TM1ValString (hPool, "cotton candy", 0)); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "sweet cotton candy",0), TM1TypeElementConsolidated()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hElement = TM1ObjectListHandleByNameGet (hPool, hDimension, TM1DimensionElements(), TM1ValString (hPool, "sweet cotton candy", 0)); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "grape",0), TM1TypeElementSimple()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); </pre>

Item	Description
Example (cont.)	<pre> hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "cherry",0), TM1TypeElementSimple()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hReturn = TM1DimensionCheck (hPool, hDimension); hDimension = TM1ObjectRegister (hPool, hServer, hDimension, TM1ValString (hPool, "APIExample", 0)); </pre>
	Out of order: For example, you can define the elements in this order: hElementcotton candy has children sour cotton candy and sweet cotton candy.hElement sour cotton candy has children green apple and lemon.hElement sweet cotton candy has children grape and cherry.
Example (cont.)	<pre> TM1P hPool; TM1V hDimension, hElementAfter, hElement, hComponent, hReturn; //get your own user and server handles hPool = TM1ValPoolCreate (hUser); hDimension = TM1DimensionCreateEmpty (hPool, hServer); hElementAfter = TM1ObjectNull (); hElement = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "cotton candy",0), TM1TypeElementConsolidated()); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "sour cotton candy",0), TM1TypeElementConsolidated()); </pre>

Item	Description
	<pre> hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "sweet cotton candy",0), TM1TypeElementConsolidated()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hElement = TM1ObjectListHandleByNameGet (hPool, hDimension, TM1DimensionElements(), TM1ValString (hPool, "sour cotton candy", 0)); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "green apple",0), TM1TypeElementSimple()); </pre>
Example (cont.)	<pre> hReturn = TM1DimensionElementComponentAdd (hPool,hElement, hComponent, 1); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "lemon",0), TM1TypeElementSimple()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hElement = TM1ObjectListHandleByNameGet (hPool, hDimension, TM1DimensionElements(), TM1ValString (hPool, "sweet cotton candy",0)); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "grape",0), TM1TypeElementSimple()); </pre>
	<pre> hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hComponent = TM1DimensionElementInsert (hPool, hDimension, hElement, TM1ValString (hPool, "cherry",0), TM1TypeElementSimple()); hReturn = TM1DimensionElementComponentAdd (hPool, hElement, hComponent, 1); hReturn = TM1DimensionCheck (hPool, hDimension); hDimension = TM1ObjectRegister (hPool, hServer, hDimension, TM1ValString (hPool, "APIExample", 0)); </pre>

TM1DimensionElementListByIndexGet

Item	Description
Purpose	Returns a list of elements by an index.
Definition	TM1V vElementList = T1V hDimension, TM1V beginIndex, TM1V nCount, TM1V iFlag)
Parameters	hDimension:a dimension handle beginIndex:a start index from where the elements are to be retrieved nCountnumber of elements to be retrieved iFlag;a flag to indicate which set of properties to be retrieved for elements.
Result	vElementList: an array of elements. The array size reflects total number of elements returned, each array include element handle, name, etc depending on the flag Array (): Array(DEL_HANDLE): Array(DEL_NAME): Array(DEL_INDEX): Array(DEL_LEVEL): Array(DEL_TYPE): Array(DEL_WEIGHT): Array(DEL_NUMBERATTRIBUTES): Array(DEL_ATTRIBUTEDATA): Array():Attribute Data Structure - list of attributes Array(AD_TYPE): Array(AD_VALUE):

TM1DimensionElementListByNamesGet

Item	Description
Purpose	Returns a list of element names in a dimension.
Definition	TM1V vElementList = TM1DimensionElementListByNamesGet(TM1V hDimension, TM1V vElemNames, TM1V iFlag)

Item	Description
Parameters	hDimension is a dimension handle vElemNames is an array of element names iFlag is a flag to indicate which set of properties to be retrieved for elements. (DEL_GET_WEIGHT DEL_GET_NUMBERATTRIBUTES DEL_GET_ATTRIBUTEDATA)
Result	vElementList is an array of elements, each array include element handle, name, etc depending on the flag Array (): Array(DEL_HANDLE): Array(DEL_NAME): Array(DEL_INDEX): Array(DEL_LEVEL): Array(DEL_TYPE): Array(DEL_WEIGHT): Array(DEL_NUMBERATTRIBUTES): Array(DEL_ATTRIBUTEDATA): Array():Attribute Data Structure - list of attributes Array(AD_TYPE): Array(AD_VALUE):

TM1DimensionUpdate

Item	Description
Purpose	Replaces a registered dimension with a new one and updates all associated cubes.
Definition	<pre>TM1IMPORT TM1V TM1API TM1DimensionUpdate(TM1PhPool, TM1V hOldDimension, TM1V hNewDimension);</pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hOldDimension is a handle of the registered dimension to be replaced. hNewDimension is a handle to the dimension that replaces the old one. The new dimension must be checked with the function TM1DimensionCheck before you call TM1DimensionUpdate.
Result	The function returns a TM1_Object. The function returns a handle to the updated dimension if the operation is successful. The old dimension is destroyed and replaced with the new one. All affected cubes are updated accordingly.
Security	The client must have ADMIN rights to the dimension being updated.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorDimensionNotChecked TM1ErrorObjectSecurityNoAdminRights
See Also	TM1DimensionCheck

TM1ElementCompomentsGet

Item	Description
Purpose	Returns an array of total child elements.
Definition	<pre>TM1V vElements = TM1ElementCompomentsGet (TM1V hParentElement, TM1V vAliasName, TM1V iStartIndex, TM1V nCount)</pre>
Parameters	hParentElement:element handle of the parent consolidated element vAliasName:name of an alias to use. NULL if no alias is to be used. iStartIndex:the start index of the child element. The index is 1-based, and begins with 1 for the first child. nCount:number of children to fetch.

Item	Description
Result	vElements:an Array containing total child elements. Each element is an array containing element handle, name, and other properties: Array(): Array(EC_HANDLE): handle to a component child. Array(EC_NAME):component element name or alias name if alias is used for the child handle. Array(EC_DIMELEMENTHANDLE): the corresponding dimension element handle based on the child handle name. Array(EC_SECURITYOWNER): security owner of the corresponding dimension element Array(EC_TYPE):element type Array(EC_COMPONENTWEIGHT): component weight of the child handle off the parent element. Array(EC_LEVEL): component element level Array(EC_INDEX): element index in the dimension

TM1GetSubsetByHandle

Item	Description
Purpose	Returns a list of the properties of a subset.
Definition	TM1V vSubsetInfo = TM1GetSubsetByHandle (TM1VhSubset)
Parameters	hSubset: a subset handle
Result	vSubsetInfo: an array containing properties of the subset. The return is the same as for “TM1SubsetListByNamesGet” on page 167 and “TM1SubsetListGet” on page 166

TM1GetViewByName

Item	Description
Purpose	Returns a view by name
Definition	TM1V sServerName, TM1V sCubeName, TM1V sViewName, TM1V bIsPrivate, TM1V iFlag

Item	Description
Parameters	sServer: server name sCubeName: cube name sViewName: View name bIsPrivate: true for private view, false for public view iFlat: TitleDimensions/ColumnDimensions/RowDimensions/CellValues, they are defined in <tm1_r7s/EnhApiConst.h as:
Result	vViewInfo: a structured Array containing view meta data and/or cell data information: Array(GV_PARENTCUBEHANDLE): handle to the cube the view belongs to Array(GV_VIEWHANDLE): unregistered handle to a copy of the view object Array(GV_SERVERHANDLE): server handle Array(GV_SERVERNAME): name of the server Array(GV_CUBENAME): name of the cube the view belongs to Array(GV_CUBEISVIRTUAL): Boolean value to indicate whether the cube is virtual Array(GV_VIEWNAME): view name Array(GV_VIEWISPRIVATE): Boolean value whether the view is private or not Array(GV_SUPPRESSZEROS): int value whether the view is zero suppressed on column row both Array(GV_SHOWAUTOMATICALLY): Boolean value whether view is automatically Recalculated Array(GV_SYNCDIMENSIONS): Array(GV_VIEWWINDOWRECT): Array(GV_VIEWFORMATSTRING): View cell value format string

Item	Description
	Array(GV_MINCOLWIDTH): Array(GV_TITLEELEMENTDATA): Selected title elements. Array (): Subset Data Structure - Title Subset Array(SD_DIMHANDLE): Array(SD_SUBHANDLE): Array(SD_DIMINDEX): Array(SD_DIMNAME): Array(SD_DIMORSUB): Array(SD_SUBALIAS): Array(SD_SUBNAME): Array(SD_ELEMENTDATA): Array(): Element Data Structure Array(ED_INDEX): Array(ED_NAME): Array(ED_ALIAS): Array(ED_FORMATSTRING): Array(ED_ISPSEUDO): Array(ED_DISPLAYPLUS): Array(ED_DISPLAYMINUS): Array(ED_DISPLAYLEVEL): Array(ED_TYPE): Array(ED_ISUPDATABLE): Array(ED_HANDLE):

Item	Description
	Array(GV_TITLEDIMENSIONDATA): Array (): Subset Data Structure - Title Subset ... Array(GV_COLUMNNDIMENSIONDATA): Array (): Subset Data Structure - Column Subset ... Array(GV_ROWNDIMENSIONDATA): Array (): Subset Data Structure - Row Subset ... Array(GV_CELLDATA): raw value of view cells from the view array. Array(iRowNum): Array of cells for a given row number. Array(iColNum): raw value of a cell at a given column number of the row.

TM1GetViewByHandle

Item	Description
Purpose	Returns a view by handle.
Definition	TM1V hView, TM1V iFlag
Parameters	hServer: a view handle Flat: same as for TM1GetViewByName.
Result	Same as for TM1GetViewByName

TM1GroupAdd

Item	Description
Purpose	Adds a new group to a server.
Definition	<pre>TM1IMPORT TM1V TM1API TM1GroupAdd(TM1P hPool, TM1V hServer, TM1V sGroupName);</pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server to which the group will be added. sGroupName is a string containing the name of the group to be added.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. This function creates a new group on the IBM Cognos TM1 server. To add clients to the new group, call TM1ClientGroupAssign.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorGroupAlreadyExists TM1ErrorGroupMaximumNumberExceeded TM1ErrorObjectSecurityNoAdminRights
See Also	TM1ClientGroupAssign Other TM1Client functions.

TM1ObjectAttributeDelete

Item	Description
Purpose	Deletes an attribute from an object and its siblings.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectAttributeDelete(TM1PhPool, TM1V hObject, TM1V hAttribute);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle of the object from which the attribute is be deleted. hAttribute is a handle of the attribute to be deleted.
Result	The function returns a TM1V containing TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. The attribute is deleted from the object and its siblings.
Security	The user must have ADMIN rights to the parent of the object.
Errors	TM1ErrorObjectAttributeDoesNotExist TM1ErrorObjectIsSecurityNoAdminRights TM1ErrorObjectIsUnregistered

Item	Description
See Also	All TM1Object functions.

TM1ObjectAttributeInsert

Item	Description
Purpose	Inserts an attribute in an object and its siblings. Also used to create an alias attribute for an object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectAttributeInsert(TM1P hPool, TM1V hObject, TM1V hAttributeBefore, TM1V sName, TM1VvType);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the registered object for which the attribute is to be created. Use TM1ServerProperty to get a registered object. For example, use TM1ServerDimensions to get a Dimension, TM1ServerCubes to get a Cube. hAttributeBefore is a handle to the attribute before which the new attribute is to be inserted. If the handle is TM1ObjectNull, the new attribute is inserted after the last attribute in the list. sName is a TM1V containing a string that specifies the name of the attribute. vType is an integer specifying the type of attribute, and which can be one of the following: TM1TypeAttributeNumeric() TM1TypeAttributeString() TM1TypeAttributeAlias()
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. The new attribute is created for the object and its siblings.
Security	The user must have ADMIN rights to the parent of the object.
Errors	TM1ErrorObjectAttributeAlreadyExists TM1ErrorObjectIsUnregistered TM1ErrorObjectIsSecurityNoAdminRights
See Also	All TM1Object functions.

TM1ObjectAttributeValueGet

Item	Description
Purpose	Retrieves the value of an attribute for any object. Also used to retrieve the value of an alias for the object specified.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectAttributeValueGet(TM1P hPool, TM1V hObject, TM1V hAttribute);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is an object handle. hAttribute is an attribute handle for the object. The possible attribute types are numeric, text, and alias, depending on the attribute you point to.
Result	Returns the value of the attribute for the object. The type of the value depends on type of the attribute.
Security	The client must have read access to the object in question in order to receive a result.
Errors	<pre>TM1ErrorObjectAttributeNotDefined TM1ErrorObjectSecurityNoReadRights TM1ObjectAttributeInsert</pre>
See Also	TM1ObjectAttributeValueSet

TM1ObjectAttributeValueSet

Item	Description
Purpose	Updates the value of an object attribute. Also used to assign a name to the alias.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectAttributeValueSet (TM1P hPool, TM1V hObject, TM1V hAttribute, TM1V vValue);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a valid object handle. hAttribute is a valid Attribute handle for the object. The possible attribute types are numeric, text, and alias, depending on the attribute you point to. vValue is the value to be assigned to the attribute. The type of value depends on the type of the attribute that you are setting.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. An alias name must not be assigned for more than one object. More than one alias name, however, may be assigned to the same object.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectAttributeValueNotDefined TM1ErrorObjectAttributeTypeConflict TM1ErrorObjectAttributeAliasConflict
See Also	TM1ObjectAttributeValueGet TM1ObjectAttributeInsert TM1ObjectAttributeValueSet

TM1ObjectCopy

Item	Description
Purpose	Copies an object from one server to another.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectCopy(TM1P hPool, TM1V hSourceObject, TM1V hDestinationObject);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSourceObject is a handle to the object to be copied. hDestinationObject is a handle to an empty object handle on the destination server.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. This function is used during replication to copy cube and dimension data from one server to another. Typically, the hSourceObject and the hDestinationObject are on different servers. The destination object is an empty object handle of the same type as the source object. It must be an unregistered object.
Security	None.
Errors	None.

Item	Description
See Also	TM1ObjectDuplicate TM1CubeCreate TM1CubePerspectiveCreate TM1DimensionCreateEmpty TM1RuleCreateEmpty TM1SubsetCreateEmpty TM1ViewCreate

TM1ObjectDelete

Item	Description
Purpose	Deletes a registered object from a server and releases its space.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectDelete(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be deleted.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. This function applies to all registered objects. The object is removed from the appropriate object list of its parent. The storage used by the object is released. All subsequent references using a handle to the object will result in the error: TM1ErrorObjectNotFound. Other errors are also possible.
Security	Client must have ADMIN privileges to the parent object.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorObjectSecurityNoAdminRights TM1ErrorObjectNotFound
See Also	TM1ObjectDestroy

TM1ObjectDestroy

Item	Description
Purpose	Destroys an unregistered object and releases its space.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectDestroy(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be destroyed.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. This function applies only to major objects that are unregistered. The storage used by the object is released. Subsequent references using the handle of the object will yield unpredictable erroneous results.
Errors	TM1ErrorObjectIsRegistered TM1ErrorObjectFunctionDoesNotApply TM1ErrorObjectNotFound TM1ErrorObjectBeingUsedByObject
See Also	TM1ObjectDelete

TM1ObjectDuplicate

Item	Description
Purpose	Makes a copy of an object in the same server or on a different server.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectDuplicate(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be copied.
Result	The function returns a handle to the copy of the object.
Security	The client must have READ rights to the object to be copied.
Errors	TM1ErrorObjectFunctionDoesNotApply> TM1ErrorObjectSecurityNoReadRights
See Also	TM1ObjectCopy

TM1ObjectFileDelete

Item	Description
Purpose	Deletes the file of a given object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectFileDelete(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose file is to be deleted.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights
See Also	Other TM1ObjectFile functions.

TM1ObjectFileLoad

Item	Description
Purpose	Used to load an object that has been unloaded.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectFileLoad(TM1P hPool, TM1V hServer, TM1V hParent, TM1V iObjectType, TM1VsObjectName);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server on which the file resides. hParent is a handle to the parent of the object whose file you want to load. iObjectType is an IBM Cognos TM1 object type, as defined in the header file Tm1api.h. For example, if the object is a cube, set this argument to TM1TypeCube(). If it is a dimension, set this argument to TM1TypeDimension(). sObjectName is the string name of the object.

Item	Description
Result	The function returns a handle to the registered object that is created when the file is loaded. The parent must be a registered object. The file to load must correspond to an object that is already registered in the server, but has been unloaded. You cannot put a file into the DB and attempt to load it.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights TM1ErrorObjectFileNotFound
See Also	Other TM1ObjectFile functions.

TM1ObjectFileSave

Item	Description
Purpose	Saves objects after significant changes are made or new objects created.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectFileSave(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose file is to be saved.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the capsule. Cube and Dimension objects are saved to the directory from which they came. If a previous version of the file is not found in any of the server directories, it is saved in the first one. Minor objects, such as hierarchies or views, are saved in the directory where their parent object resides.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights
See Also	Other TM1ObjectFile functions.

TM1ObjectListCountGet

Item	Description
Purpose	Retrieves the number of items in a list property.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectListCountGet(TM1P hPool, TM1V hObject, TM1V iPropertyList);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose list property is being queried. iPropertyList is a list property index value for the object. The IBM Cognos TM1 API supplies functions that return this index. For example, to retrieve the number of dimensions in a cube, set this variable equal to TM1CubeDimensions(). Other property index values are listed in Tm1api.h.
Result	This function returns a TM1V value containing a TM1_INDEX. This index contains the number of items on the list. Use TM1ValIndexGet to retrieve the data. This function applies to all objects.
Security	The client must have READ rights to the object.
Errors	<pre>TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList</pre>
See Also	Other TM1ObjectList functions.

TM1ObjectListHandleByIndexGet

Item	Description
Purpose	Retrieves an item on a list property given an index.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectListHandleByIndexGet(TM1P hPool, TM1V hObject, TM1V iPropertyList, TM1V iIndex);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose list property is being queried. iPropertyList is a list property index value for the object. The IBM Cognos TM1 API supplies functions that return this index. For example, to retrieve a dimension handle from a cube, set this variable equal to TM1CubeDimensions(). Other property index values are listed in Tm1api.h. iIndex is the index of the item within the list.

Item	Description
Result	The function returns a handle to the requested item. This function applies to all objects.
Security	The client must have READ rights to the object.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectList functions.

TM1ObjectListHandleByNameGet

Item	Description
Purpose	Retrieves an item in a list property given a name.
Definition	<pre> TM1IMPORT TM1V TM1API TM1ObjectListHandleByNameGet(TM1P hPool, TM1V hObject, TM1V iPropertyList, TM1V sName); </pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose list property is being queried. iPropertyList is a list property index value for the object. The IBM Cognos TM1 API supplies functions that return this index. For example, to retrieve a dimension handle from a cube, set this variable equal to TM1CubeDimensions(). Other property index values are listed in Tm1api.h. sName is a string containing the name of the requested object.
Result	The function returns the handle of the requested object. This function applies to all TM1 objects except subsets. To retrieve the elements in a subset, use the function TM1ObjectListHandleByIndexGet or call TM1ObjectListHandleByNameGet passing the property TM1DimensionElements(). In summary: <pre> TM1ObjectListHandleByNameGet (hPool, hDimensionObject, TM1DimensionElements(), vsName); // This function works TM1ObjectListHandleByIndexGet (hPool, hSubsetObject, TM1SubsetElements(), vsName); // This function works TM1ObjectListHandleByNameGet (hPool, hSubsetObject, TM1SubsetElements(), vsName); // This function returns an error </pre>

Item	Description
Security	The client must have READ rights to the object.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectList functions.

TM1ObjectPrivateDelete

Item	Description
Purpose	Deletes a previously registered private object.
Definition	TM1IMPORT TM1V TM1API TM1 (TM1P hPool, TM1V hObject);
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject a handle for the private object you want to delete.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. The object is removed from the appropriate object list of its parent. The storage used by the object is released. All subsequent references using a handle to the object will result in the error: TM1ErrorObjectDeleted
Security	You are only allowed to delete private objects that you have created.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorObjectSecurityNoAdminRights TM1ErrorObjectDeleted
See Also	TM1ObjectPrivateRegister TM1ObjectDestroy

TM1ObjectPrivateListCountGet

Item	Description
Purpose	Returns the number of items in the list property of a private object.
Definition	TM1IMPORT TM1V TM1API TM1ObjectPrivateListCountGet(TM1P hPool, TM1V hObject, TM1V iPropertyList);

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose list property is being queried. It is always a parent handle. iPropertyList is a constant defined in tm1api.h. It is always a list of children. These values are returned by the object property value functions supplied by the API. For example, the constant TM1ObjectList returns a property index for the list property of an object. If hObject is a server handle and iPropertyList is TM1ServerDimensions, this function returns the number of private dimensions on the server.
Result	Returns a value containing the number of items on the list.
Security	None.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectPrivate functions.

TM1ObjectPrivateListHandleByIndexGet

Item	Description
Purpose	Given an index, this function returns the handle of the object in that position of a list property.
Definition	<pre>TM1IMPORT TM1V TM1API ObjectPrivateListHandleByIndexGet(TM1P hPool, TM1V hObject, TM1V iPropertyList, TM1V iIndex);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose list property is being queried. It is always a parent handle. iPropertyList is a constant defined in tm1api.h. These values are returned by the object property value functions supplied by the API. iIndex is the index of the item within the list. For example, the constant TM1ObjectList returns a property index for the list property of an object. If hObject is a server handle and iPropertyList is TM1ServerDimensions, this function returns the handle of the dimension in the iIndex position on the server.
Result	The function returns a handle to the requested item. This function used only to locate private sub-objects of shared objects.

Item	Description
Security	None.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectPrivate functions.

TM1ObjectPrivateListHandleByNameGet

Item	Description
Purpose	Returns a handle to an object, provided that the object name is on the list.
Definition	<pre> TM1IMPORT TM1V TM1API ObjectPrivateListHandleByNameGet(TM1P hPool, TM1V hObject, TM1V iPropertyList, TM1V sName); </pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object whose list property is being queried. iPropertyList is a constant defined in tm1api.h. These values are returned by the object property value functions supplied by the API. sName is a string containing the name of the requested object. For example, the constant TM1ObjectList returns a property index for the list property of an object. If hObject is a server handle and iPropertyList is TM1ServerDimensions, this function returns the handle of the sName of the dimension on the server.
Result	The function returns the handle of the requested object. This function is used only to locate private sub-objects of shared objects.
Security	None.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectPrivate functions.

TM1ObjectPrivatePublish

Item	Description
Purpose	Makes a private object into a public (or shared) object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectPrivatePublish(TM1P hPool, TM1V hObject, TM1V sName);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object that is being published. sName the name by which other applications can access the object.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. This function adds the name specified by sName to the list property of the parent of the object specified by hObject. This function makes a public copy of the object and assigns it a new name. All sub-objects must be public, otherwise the function will fail. The original private object is removed by this function, leaving only the new public object.
Security	To publish a private object, you must be a member of the ADMIN group.
Errors	TM1ErrorViewHasPrivateSubsets
See Also	TM1ObjectPrivateRegister

TM1ObjectPrivateRegister

Item	Description
Purpose	Registers a private object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1 ObjectPrivateRegister (TM1P hPool, TM1V hParent, TM1V hObject, TM1V sName);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hParent is the handle of the parent of the object you want to register. hObject is an object handle to the private object you want to register. sName is a TM1V containing a TM1_STRING. This string is the name under which you register the object. Applications can retrieve the object by submitting this name to the function TM1ObjectPrivateListHandleByNameGet.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. This function assigns a name to an object, makes it a private sub-object of its parent, and stores it permanently. Private objects can have the same name as shared objects, but this practice is not recommended.
Security	The creator of a private object has ADMIN rights to it.
Errors	None.
See Also	TM1ObjectRegister TM1ObjectPrivatePublish TM1PrivateListHandle functions

TM1ObjectPropertyGet

Item	Description
Purpose	Retrieves the value of a property for an object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectPropertyGet(TM1P hPool, TM1V hObject, TM1V vProperty);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a valid object handle. vProperty is a property index value for the object. These values are returned by the object property value functions supplied by the API. For example, these two lines return a string containing the name of the object: <pre>vsObjectName = TM1ObjectPropertyGet(pGeneral, hObject, TM1ObjectName()); sHierarchyName = TM1ValStringGet(hUser, vsObjectName);</pre>
Result	The function normally returns the value of the property for the object. The type of the value depends on the property, and could be any of the standard IBM Cognos TM1 types. This function does not work for list properties. List properties must be handled using the TM1ObjectList functions. This function applies to all objects.

Item	Description
Security	The client must have READ access to the object in question in order to receive a result.
Errors	TM1ErrorObjectPropertyNotDefined TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyIsList

TM1ObjectPropertySet

Item	Description
Purpose	Updates the value of a property for an object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectPropertySet(TM1P hPool, TM1V hObject, TM1V vProperty, TM1V vValue);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a valid object handle. vProperty is a property index value for the object. These values are returned by the object property value functions supplied by the API. For example, the function TM1ObjectName() returns a string containing the name of the object. vValue is the value to be assigned to the property.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function applies to all objects. This function cannot set all properties. Some properties cannot be updated.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights TM1ErrorObjectPropertyNotDefined TM1ErrorObjectPropertyIsList
See Also	TM1ObjectPropertyGet

TM1ObjectRegister

Item	Description
Purpose	Registers an object with its parent object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectRegister(TM1P hPool, TM1V hParent, TM1V hObject, TM1V sName);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hParent is a handle of the parent object. hObject is a handle to the object to be registered. sName is the name under which the object will be registered.
Result	Returns the handle to the registered object. The object is put in the appropriate object list of the parent object. The old handle becomes invalid.
Security	The client must have ADMIN rights to the parent object.
Errors	<pre>TM1ErrorObjectSecurityNoAdminRights TM1ErrorObjectIsRegistered TM1ErrorObjectNameInvalid TM1ErrorObjectNameIsBlank TM1ErrorObjectNameExists</pre> If the object is a Dimension, error is TM1ErrorDimensionCannotBeCompiled If the object is a View, error is TM1ErrorViewHasPrivateSubsets
See Also	TM1ObjectPrivateRegister

TM1ObjectReplicate

Item	Description
Purpose	Copies an object from star server to a planet server.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectReplicate(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle of the object to be replicated. This is typically a cube handle.

Item	Description
Result	If this function is successful, it returns a Boolean 1. The data and metadata of the requested object copied from the star server to the planet server.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	TM1ObjectReplicate TM1ObjectReplicationSourceObjectName TM1CubeReplicationSyncRule TM1CubeReplicationSyncViews TM1DimensionReplicationSyncSubsets

TM1ObjectSecurityLock

Item	Description
Purpose	Permanently prohibits WRITE access to an object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectSecurityLock(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be locked.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The new restrictions take effect immediately. Only a server administrator can remove an object lock. This function applies to all objects.
Security	The client must have LOCK rights to the object.
Errors	TM1ErrorObjectSecurityNoLockRights
See Also	TM1ObjectSecurityUnLock TM1ObjectSecurityReserve

TM1ObjectSecurityRelease

Item	Description
Purpose	Allows WRITE access to an object that was previously reserved.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectSecurityRelease(TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be released.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function applies to all objects.
Security	The client must have ADMIN rights to the object, or have previously reserved the object.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityReserve

Item	Description
Purpose	Temporarily prohibits WRITE access to an object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectSecurityReserve(TM1PhPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be reserved.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The new restrictions take effect immediately. This function applies to all objects.
Security	The client must have RESERVE rights to the object.
Errors	TM1ErrorObjectSecurityNoReserveRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityRightGet

Item	Description
Purpose	Retrieves the security rights for a given object for a given group.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectSecurityRightGet(TM1P hPool, TM1V hObject, TM1V hGroup);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object. hGroup is a handle to a client or a group.
Result	The function returns an integer value indicating the current rights to the object for the group. The result will be equivalent to one of the following values: TM1SecurityRightNone() TM1SecurityRightRead() TM1SecurityRightWrite() TM1SecurityRightReserve() TM1SecurityRightLock() TM1SecurityRightAdmin() This function applies to all objects. This function is designed to allow IBM Cognos TM1 server administrators to check the access rights for clients and groups to objects on the server. If the hGroup argument is a handle to a group, the function returns the security rights for the group. If the hGroup argument is a handle to a client, the function returns the highest level of access available to that client.
Security	The client must be a member of the ADMIN group to retrieve the security for groups.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1ObjectSecurity functions.

Item	Description
Example	The following example shows how to compare against constants to get security, <pre> nReturnCode = TM1ObjectSecurityRightGet (hPool, hCube, hGroup); // error checker in here if (TM1ValIndexGet (hUser, nReturnCode) == TM1ValIndexGet (hUser, TM1SecurityRightWrite ()); // default rights </pre>

TM1ObjectSecurityRightSet

Item	Description
Purpose	Sets the security rights for a given object for a given group.
Definition	<pre> TM1IMPORT TM1V TM1API TM1ObjectSecurityRightSet(TM1P hPool, TM1V hObject, TM1V hGroup, TM1V iRight); </pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object. hGroup is a handle to the group. iRight is the rights level to be assigned, which is one of the following: TM1SecurityRightNone TM1SecurityRightRead TM1SecurityRightWrite TM1SecurityRightReserve TM1SecurityRightLock TM1SecurityRightAdmin This function applies to all objects.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The new rights take effect immediately.
Security	The client must be a member of the ADMIN group to set security for a group.

Item	Description
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityUnLock

Item	Description
Purpose	Removes a lock from a previously locked object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectSecurityUnLock (TM1P hPool, TM1V hObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle to the object to be unlocked.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function applies to all objects.
Security	The client must have ADMIN rights to the object.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ProcessExecute

Item	Description
Purpose	Executes a TurboIntegrator process on an IBM Cognos TM1 server.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ProcessExecute(TM1PhPool, TM1V hProcess, TM1V hParametersArray);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the TM1 server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hParametersArray is a value capsule containing an array of parameters. Each parameters can be a number (created with either TM1ValIndex or TM1ValReal functions) or a string (created with TM1ValString functions). This array has to match the exact definition of the process's parameters in number and type; if it doesn't an error is returned and the process is not executed. A process with no parameters takes an array of zero elements.
Result	The result TM1V object should first be checked to see if it an error item. If so, the error value may be retrieved. The error value may be one of the following values: TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectIsUnregistered TM1ErrorObjectInvalid If the return is not an error object is should be a Boolean object. If the Boolean is 0, the process execution generated errors. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. If you need more specific information about the error resulting from the process, call the function TM1ProcessExecuteEx in place of this function.
Security	None
Errors	As described above, the function may return one of the following error codes: TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectIsUnregistered TM1ErrorObjectInvalid The function writes all error messages to an error log file in the TM1 server's data directory. The error log file name is the same as the process, with a time stamp appended.
See Also	TM1ChoreExecute

TM1ProcessExecuteEx

Item	Description
Purpose	Executes a TurboIntegrator process on an IBM Cognos TM1 server.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ProcessExecute(TM1PhPool, TM1V hProcess, TM1V hParametersArray);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the TM1 server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hParametersArray is a value capsule containing an array of parameters. Each parameters can be a number (created with either TM1ValIndex or TM1ValReal functions) or a string (created with TM1ValString functions). This array has to match the exact definition of the process's parameters in number and type; if it doesn't an error is returned and the process is not executed. A process with no parameters takes an array of zero elements.
Result	The result TM1V object should first be checked to see if it an error item. If so, the error value may be retrieved. The error value may be one of the following values: • TM1ErrorObjectSecurityNoReadRights • TM1ErrorObjectIsUnregistered • TM1ErrorObjectInvalid If the return is not an error object is should contain a TM1 array. The array contains two elements. The first element is an error code. The error codes are listed below. The second element is the path to the error log file. The error log file is generated only if an error occurs. Returns an index of 1 or the object if the execution returned was normal or "process break."
Security	None
Errors	The returned array contains 0 if the process was successful or one of the following error codes. <pre>TM1ProcessAborted TM1ProcessHasMinorErrors TM1ProcessQuitCalled TM1ProcessCompletedWithMessages</pre>
See Also	TM1ChoreExecute

TM1ProcessExecuteSQLQuery

Item	Description
Purpose	Opens a connection to an SQL da source. Builds and returns an array of records selected via the passed SQL query.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ProcessExecuteSQLQuery(TM1P hPool, TM1V hProcess, TM1V hParametersArray);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the IBM Cognos TM1 server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hParametersArray is a value capsule containing an array of parameters as follows: 1: DSN Name 2: User Name 3: Password 4: SQL Statement 5: (optional) Limit on number of records from query
Result	Returns an array of the records selected by the passed query or an error code.
Security	None
Errors	The returned array contains one of the following error codes. - er_DatabaseInfoIncomplete - er_DatabaseConnectionFailed - er_DatabaseQueryExecutionFailed

TM1ProcessVariableNameIsValid

Item	Description
Purpose	Tests whether a process variable name is valid in the specified IBM Cognos TM1 process.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ProcessVariableNameIsValid(TM1PhPool, TM1V hProcess, TM1V hVariableName);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the TM1 server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hVariableName is a string value containing the process variable name.
Result	Returns an array containing bool and possibly error messages.
Security	None
Errors	The returned array contains one of the following error codes.

TM1RuleAttach

Item	Description
Purpose	Attaches a rule to a cube.
Definition	<pre>TM1IMPORT TM1V TM1API TM1RuleAttach(TM1P hPool, TM1V hRule);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hRule is a handle to a rule.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The function installs the rule as a property of its parent cube. The name of the property is TM1CubeRule.
Security	You must have ADMIN rights to the parent cube.
Errors	None.
See Also	Other TM1Rule functions.

TM1RuleCheck

Item	Description
Purpose	Checks a rule for syntax.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1RuleCheck(TM1P hPool, TM1V hRule);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hRule is a handle to a rule.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the syntax of the rule is correct. If the Boolean is 0, a syntax error was detected in the rule. Use TM1ValBoolGet to retrieve the Boolean from the value capsule. If the rule has a syntax error, you can retrieve the line containing the error by calling TM1ObjectPropertyGet for the rule properties TM1RuleErrorLine and TM1RuleErrorString.
Security	You must have READ access to the rule object.
Errors	None.
See Also	Other TM1Rule functions.

TM1RuleCreateEmpty

Item	Description
Purpose	Creates an empty rule, and returns a handle to that rule.
Definition	<pre>TM1IMPORT TM1V TM1API TM1RuleCreateEmpty(TM1P hPool, TM1V hCube, TM1V hType);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hCube a handle to the cube to which the rule applies. hType is a handle to a value. Set this variable equal to TM1TypeRuleCalculation() for a calculation rule, and TM1TypeRuleDrill() for a drilldown rule.
Result	Returns a handle to an empty rule object. You can add lines to the rule object by calling TM1RuleLineInsert. You can compile a rule using TM1RuleCheck. Rules do not require registration, but must be attached to a cube with the function TM1RuleAttach.
Security	None.
Errors	None.

Item	Description
See Also	TM1RuleLineInsert TM1RuleCheck Other TM1Rule functions.

TM1RuleDetach

Item	Description
Purpose	Detaches a rule from a cube.
Definition	<pre>TM1IMPORT TM1V TM1API TM1RuleDetach(TM1PhPool, TM1V hRule);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hRule is a handle to a rule.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. This function deletes the rule from the TM1CubeRule property of the parent cube.
Security	You must have ADMIN rights to the parent cube.
Errors	None.
See Also	Other TM1Rule functions.

TM1RuleLineGet

Item	Description
Purpose	Retrieves a line from a rule.
Definition	<pre>TM1IMPORT TM1V TM1API TM1RuleLineGet(TM1P hPool, TM1V hRule, TM1V iPosition);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hRule is a handle to a rule. iPosition is a TM1V containing a TM1_INDEX. This value indicates which line you want to retrieve from the rule. To retrieve the first line of the rule, set this value to 1.

Item	Description
Result	Returns a TM1V containing a TM1_STRING. Use TM1ValStringGet to retrieve information. The string contains a single line of the rule specified by hRule.
Security	You must have WRITE access to the parent cube.
Errors	None.
See Also	TM1RuleLineInsert TM1RuleCheck Other TM1Rule functions.

TM1RuleLineInsert

Item	Description
Purpose	Inserts a line into a rule.
Definition	<pre>TM1IMPORT TM1V TM1API TM1RuleLineInsert(TM1P hPool, TM1V hRule, TM1V iPosition, TM1V sLine);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hRule is a handle to a rule. A handle to a new rule is generated by the function TM1RuleCreate. iPosition is a TM1V containing a TM1_INDEX. This value indicates the position at which the line will be inserted within the rule. To insert this line at the beginning of the rule, set this value to 1. sLine is a string containing the line to add to the rule.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation is successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The function adds a single line of the rule specified by hRule.
Security	None.
Errors	TM1ErrorFailedToInsertLine TM1ErrorObjectNotFound
See Also	Other TM1Rule functions.

TM1ServerBatchUpdateFinish

Item	Description
Purpose	Disables IBM Cognos TM1 batch updates. Commits or discards data entered while batch update is enabled.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ServerBatchUpdateFinish(TM1P hPool, TM1V hServer, TM1V bDiscard);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a TM1 server handle. This handle is returned by the function TM1SystemServerConnect. bDiscard is a TM1V containing a boolean. If the boolean is TRUE, all cell changes that occurred while batch update mode was enabled will be discarded. If the boolean is FALSE, all cell changes will be applied to the cubes. Calculations involving changed cubes are invalidated.
Result	This function disables batch update mode, and either applies or discards the cell changes that were made while batch update mode was enabled. Batch updates allow you to improve the performance of input-intensive applications by holding changes to cube data and saving those changes to cubes in a single batch. When you initiate batch updates by calling TM1ServerBatchUpdateStart, TM1 creates a temporary storage structure on the target server. All edits to cubes for that server are held in the temporary storage structure until you call TM1ServerBatchUpdateFinish. When you call TM1ServerBatchUpdateFinish, all edits held in temporary storage are either committed or destroyed, depending on the setting of the bDiscard flag. The temporary storage structure is destroyed. By default, batch update is disabled on an IBM Cognos TM1 server.
Security	None.
Errors	None.
See Also	TM1ServerBatchUpdateStart

TM1ServerBatchUpdateIsActive

Item	Description
Purpose	Returns a boolean TRUE if IBM Cognos TM1 batch update mode is enabled.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1ServerBatchUpdateIsActive(TM1P hPool, TM1V hServer);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a TM1 server handle. This handle is returned by the function TM1SystemServerConnect.
Result	This function returns TRUE if batch update mode is enabled. It returns FALSE if batch update mode is disabled.
Security	None.
Errors	None.
See Also	TM1ServerBatchUpdateStart TM1ServerBatchUpdateFinish

TM1ServerBatchUpdateStart

Item	Description
Purpose	Enables IBM Cognos TM1 batch updates.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ServerBatchUpdateStart(TM1P hPool, TM1V hServer);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is an IBM Cognos TM1 server handle. This handle is returned by the function TM1SystemServerConnect.
Result	This function enables batch update mode. Batch updates allow you to improve the performance of input-intensive applications by holding changes to cube data and saving those changes to cubes in a single batch. When you initiate batch updates, TM1 creates a temporary storage structure on the target server. All edits to cubes residing on the server are held in the temporary storage structure until you call TM1ServerBatchUpdateFinish. By default, batch update is disabled on a TM1 server. TM1 data spreading is disabled while batch update mode is enabled.
Security	None.
Errors	None.
See Also	TM1ServerBatchUpdateFinish

TM1ServerDimensionListGet

Item	Description
Purpose	Returns a list of dimensions by cube.
Definition	TM1V vDimensionList = TM1CubeDimensionListGet(TM1V hCube)
Parameters	hServer is a cube handle
Result	vDimensionList is an array of dimensions of the specified cube in exact order they exist in the cube, each array include dimension handle, name, element count, public subset count, private subset count, etc. Note the(index + 1) of top level array should match the index of dimension in the cube (1-based index in IBM Cognos TM1 server). Array ();// array (index+1)matching dimension index in the cube Array(CDL_DIMENSIONHANDLE): Array(CDL_DIMENSIONNAME): Array(CDL_EIEMENTCOUNT): Array(CDL_LASTUPDATETIME): Array(CDL_NUMBEROFLEVELS):

TM1ServerDimensionListByNamesGet

Item	Description
Purpose	Returns a list of dimensions by server.
Definition	TM1V vDimensionList = TM1ServerDimensionListByNamesGet(TM1V hServer, TM1V vDimensionNames)
Parameters	hServer is a server handle vDimensinNames is an array of dimension names

Item	Description
Result	vDimensionList is an array of dimensions of the specified server, each array include dimension handle, name, element count, last update time, etc. Array (): Array(SDL_DIMENSIONHANDLE): Array(SDL_DIMENSIONNAME): Array(SDL_ELEMENTCOUNT): Array(SDL_LEVELSCOUNT): Array(SDL_LASTUPDATETIME): ...

TM1ServerDisableBulkLoadMode

Item	Description
Purpose	Turns off bulk load mode operation.
Definition	TM1V TM1ServerDisableBulkLoadMode(TM1P hPool, TM1Server hServer)
Parameters	TM1P is a pool handle obtained with TM1ValPoolCreate TM1Server is a handle to the current server.
See Also	For more details, see the "Using Bulk Load Mode" section in the IBM Cognos TM1 <i>TurboIntegrator Guide</i>

TM1ServerEnableBulkLoadMode

Item	Description
Purpose	Turns on bulk load mode operation.
Definition	TM1V TM1ServerEnableBulkLoadMode(TM1P hPool, TM1Server hServer)
Parameters	TM1P is a pool handle obtained with TM1ValPoolCreate TM1Server is a handle to the current server.
See Also	For more details, see the "Using Bulk Load Mode" section in the IBM Cognos TM1 <i>TurboIntegrator Guide</i>

TM1ServerLogClose

Item	Description
Purpose	Terminates access to a server's log file.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ServerLogClose (TM1P hPool, TM1V hServer);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer a handle to a server object.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	You must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerLogNext

Item	Description
Purpose	Retrieves the next data item from a log file.
Definition	<pre>TM1IMPORT TM1V TM1API T M1ServerLogNext(TM1P hPool, TM1V hServer);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer a handle to a server object.

Item	Description
Result	Returns the next item (field) in the log file. The fields in a log record are as follows: Date/time of the change (string YYYYMMDDhhmmss GMT). Client performing the change (string) Transaction type (string) Old value (string or real) New value (string or real) Name of Cube changed (string) Dimension elements (from two to sixteen) (string). Boolean 0 to indicate the end of the record A Boolean 0 subsequent to the end of the last record indicates end of the log file has been reached. Note that access can be resumed after more records have been written to the log, without closing and re-opening the log. A Boolean 1 indicates that the returned item is identical to the corresponding one in the previous record.
Security	You must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerLogOpen

Item	Description
Purpose	Starts access to a server's log file.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ServerLogOpen (TM1P hPool, TM1V hServer, TM1V sStartTime, TM1V sCubeFilter, sUserFilter, sFlagFilter);</pre>

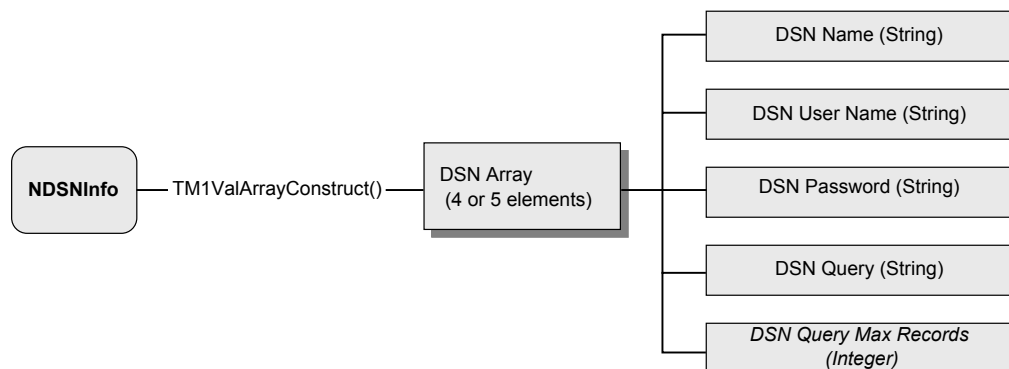
Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to a server object. sStartTime is the time stamp (GMT) after which the log records are to be retrieved. The time stamp is written as a numeric string of the form: YYYYMMDDhhmmss. sCubeFilter is a string pattern to match. A * indicates that you want to retrieve all records after the start time. You can also use * for sUserFilter and sFlagFilter. sUserFilter is a string containing an IBM Cognos TM1 client name. For example, if sUserFilter = "usr2", only log records for usr2 are written to the log file. sFlagFilter is used to filter records by flag string.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The function returns the first field of the first log record with a time stamp greater than sStartTime. If there are no such records, it returns a Boolean 0.
Security	You must have ADMIN rights to the IBM Cognos TM1 server.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerOpenSQLQuery

This diagram shows the array of four elements.

Item	Description
Purpose	Executes any SQL Query from the IBM Cognos TM1 client and returns a SQL Table object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ServerOpenSQLQuery(TM1P hPool, TM1V hServer, TM1V hDsnInfo);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to a server object. hDSNInfo is TM1V containing an array of 4 elements, as shown in the diagram following the table.

Item	Description
	The DSN referred to by DSN Name must be established on the IBM Cognos TM1 server machine. The DSN Query Max Records element in the DSN Array is optional.
Result	The function returns a TM1V containing a TM1SqlTable object. Pass this object to the function TM1SqlGetNextRows to retrieve the data generated by the SQL statement. Typically, you follow this function call with a loop that calls TM1SqlGetNextRows until there are no more rows. Then, call TM1ObjectDestroy to destroy the SQL Query object. Unlike other IBM Cognos TM1 objects, SQL query objects are session-dependent. You cannot save a SQL query object on the TM1 server. There is no list of SQL query objects. When you log out, all SQL Query objects are destroyed.
Security	None.
Errors	None.
See Also	TM1SQLTableGetNextRows TM1ObjectDestroy



TM1ServerPasswordChange

Item	Description
Purpose	Changes the client's current password in a server.
Definition	<pre> TM1IMPORT TM1V TM1API TM1ServerPasswordChange(TM1P hPool, TM1V hServer, TM1V sNewPassword); </pre>

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server in which to change the password. sNewPassword is a string to be used as the new password.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The function changes the client's password.
Security	None.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerSandboxesDelete

Item	Description
Purpose	TM1ServerSandboxesDelete allows administrators to discard user sandboxes that match certain criteria. This functionality operates server side and is available through TurboIntegrator and the API function TM1ServerSandboxesDelete. Using this feature in a TurboIntegrator process, administrators can schedule maintenance using automated chores.
Definition	<pre>TM1V TM1API TM1ServerSandboxesDelete(TM1P hPool, TM1V hServer, TM1V constraints);</pre>
Parameters	An arbitrary length array of predicates. Each predicate is an array containing an attribute index, condition index, and value string. The attribute index indicates an attribute of a sandbox. The condition index indicates a condition, e.g. ">" or "=". The value string is a possible value of the attribute on which sandboxes should be conditionally filtered. The array is limited to 100 predicates.

Description

This function uses a "predicate" to describe the sandbox being deleted. A predicate can be read as: "Delete sandboxes whose *attribute* is *condition value*."

For example: "Delete sandboxes whose size is greater than 10 MB." In this example, the attribute is the "size" of the sandbox, the condition is "greater than", and the value is "10 MB".

Filter Attributes

Attribute and Description	Valid Conditions and Value Type
TM1SandboxAdminToken_UpdateDate Timestamp of the last write action performed in the sandbox.	TM1SandboxAdminToken_LessThan, TM1SandboxAdminToken_Equals, TM1SandboxAdminToken_GreaterThan Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
TM1SandboxAdminToken_AccessDate Timestamp of the last unload of a sandbox.	TM1SandboxAdminToken_LessThan, TM1SandboxAdminToken_Equals, TM1SandboxAdminToken_GreaterThan Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
TM1SandboxAdminToken_CreationDate Timestamp of the creation of a sandbox.	TM1SandboxAdminToken_LessThan, TM1SandboxAdminToken_Equals, TM1SandboxAdminToken_GreaterThan Timestamp in international standard format, i.e. YYYY-MM-DD. Days are the most granular units.
TM1SandboxAdminToken_Size The in-memory size of a sandbox.	TM1SandboxAdminToken_LessThan, TM1SandboxAdminToken_Equals, TM1SandboxAdminToken_GreaterThan Size following log4cxx's conversion rules (see configuration parameter AuditLogMaxTemp FileSize) For example, 10 MB. Kilobytes are the most granular units.
TM1SandboxAdminToken_Name The name of a sandbox.	TM1SandboxAdminToken_Equals, TM1SandboxAdminToken_Containing String.
TM1SandboxAdminToken_Client The owning client of a sandbox.	TM1SandboxAdminToken_Equals String.
TM1SandboxAdminToken_Group A group of which the owning client of a sandbox is a member.	TM1SandboxAdminToken_Equals String.

Logging and Returns

Sandbox deletion is logged using the preexisting audit logging functionality. Additionally, a more detailed report of the effects of sandbox administration is included in the debug log (tm1server.log) at INFO level. This report will include the list of affected sandboxes, as well as some of their attributes, and any errors encountered.

TM1ServerSandboxesDelete returns only a success or failure status.

Semantics

Predicate List

Multiple predicates passed in a single call to `TM1ServerSandboxesDelete` are conjunctive. In other words, for a sandbox to match the passed criteria, all predicates must be true. Multiple calls to `TM1ServerSandboxesDelete` can be used to achieve disjunctive behavior. Only one occurrence of each attribute is allowed per call to `TM1ServerSandboxesDelete`. For example, passing client twice is invalid as a sandbox has only one owning client. When multiple occurrences of an attribute are detected, a warning displays in the detailed report, however, the operation will not abort in failure. In such a case, the predicates are tested as with any other query, but the results set is always empty.

Locking

To avoid massive locking issues, `TM1ServerSandboxesDelete` looks at the sandboxes of a client as a point-in-time snapshot and then, when possible, release any locks that would ensure a serializable transaction. Because of this behavior, once a client is "passed" in the iteration of all clients, a sandbox matching the filter criteria may be added to that client before the maintenance transaction completes. This behavior is similar to the behavior that occurs when a sandbox is added to the client immediately after the transaction completes.

Scope

Members of the ADMIN (super-user) and the DataAdmin groups will have access to all sandboxes of all clients. They must explicitly specify the client attribute to limit the scope of their call to `TM1ServerSandboxesDelete` to only their own sandboxes. All other users have access to only their own sandboxes; if they specify a different client, or a group to which they do not belong, the function will abort in failure and return a privilege error.

In-Use Sandboxes

When a sandbox meets the criteria for deletion, but is currently in use, that sandbox will not be deleted. An entry will appear in the debug log info-level report indicating the occurrence.

Access and Update Dates

Date attributes can be matches with, at most, day granularity. Because of this restriction, recording of these attributes is correspondingly granular. Last Update Date is not updated on individual cell writes. Instead, the system records the unload date of a sandbox that has had something written to it while it was loaded in memory. For such sandboxes, Last Access Date and Last Update Date will be the same. Only Last Access Date is updated on the unloading of a sandbox from memory. Also, because in-memory sandboxes are not subject to `(TM1)ServerSandboxesDelete`, Last Access Date is not updated when a sandbox is loaded into memory.

For example, consider the follow usage scenario:

Table 1. Last Access Day Example

Day	Time	Action
1	1	Load Sandbox S
1	2	Write 1
2	3	Read 1

Table 1. Last Access Day Example (continued)

Day	Time	Action
2	4	Unload Sandbox

A user is working with sandbox over the course of two days (perhaps for a much shorter period encompassing the day change.) At time 4, when the sandbox is unloaded, Last Update Date is set to 2, rather than 1 where the last update actually occurred. Last Access Date is also set to 2 at time 4 in this case. If Write1 were instead a read, only Last Access Date would be set to 2, while Last Update Date wouldn't be changed.

Example

```
TM1V predicateTokens[2][3] = { { TM1SandboxAdminToken_Client(),
TM1SandboxAdminToken_Equals(), TM1ValString( { TM1SandboxAdminToken_Name(),
TM1SandboxAdminToken_Equals(), TM1ValString( m_hPool, "Best Case", 0 )

TM1V predicates[2] = { TM1ValArray( m_hPool, predicateTokens[0], 3 ), TM1ValArray(
m_hPool, predicateTokens[TM1V aPredicates( TM1ValArray( m_hPool, predicates, 2 ) );

TM1V hResult( TM1ServerSandboxesDelete( m_hPool, m_hServer, aPredicates ) );
```

TM1ServerSecurityRefresh

Item	Description
Purpose	Updates internal security structures with information from the IBM Cognos TM1 security cubes.
Definition	TM1IMPORT TM1V TM1API TM1ServerSecurityRefresh(TM1P hPool, TM1V hServer);
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to the TM1 server. This handle is returned by a successful call to the function TM1SystemServerConnect.
Result	The function reads the security information from the TM1 security cubes, and updates the TM1 server's internal security information. You should call this function whenever you make one or more changes to one of the following TM1 security cubes, and you want the security changes to take effect. The TM1 security cubes are listed below. }Application_Security }Chore_Security }Client_Groups }Client_Security }Cube_Security }Dimension_Security }Process_Security
Security	None.

Item	Description
Errors	None.

TM1SQLTableGetNextRows

Item	Description
Purpose	Returns rows of a SQL table object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SQLTableGetNextRows(TM1P hPool, TM1V hSQLTable, TM1V aColumnSelection);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSQLTable is a handle to a SQL table object. The IBM Cognos TM1 SQL Table object is created by the function TM1CubeCellDrillListGet. aColumnSelection is an array of the selected columns' name.
Result	The function returns a TM1V array which includes data of the fetched rows. Data from the specified columns is returned. You can set the number of rows by setting the SQL table object TM1SQLTableRowsetSize to the number of your choice. Set this before calling TM1ObjectPropertySet before you call TM1SqlTableGetNextRows.
Security	None.
Errors	None.
See Also	TM1CubeCellDrillListGet TM1CubeCellDrillObjectBuild

TM1SubsetAll

Item	Description
Purpose	Populates a subset with all the elements of the parent dimension.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetAll(TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle to the subset. It is obtained with TM1SubsetCreateEmpty.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean. This function is often used to populate a subset that is to contain most of the elements in the parent dimension. After using this function, you can use TM1SubsetSelectionDelete to remove the unwanted elements from the subset.
Security	None.
Errors	TM1ErrorObjectNotFound
See Also	TM1SubsetCreateEmpty

TM1SubsetCreateByExpression

Item	Description
Purpose	Creates a subset from an MDX expression.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetCreateByExpression(TM1P hPool, TM1V hServer, TM1V sExpression);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect. sExpression is a TM1V containing a string. The string is an MDX Expression that creates a subset.
Result	Creates a subset from an MDX expression. The expression itself can be created through the TM1 Subset Editor Record Expression command. The following shows an example of MDX generated by the subset editor. <pre>{TM1FILTERBYPATTERN({TM1SubsetBasis()}, "I*")}</pre> Since this is a dynamic subset, the subset will contain only those elements that meet the requirements of the MDX expression. The population of the subset can change over time as elements are added and removed from the dimension. Once you register the subset with TM1ObjectRegister, you can retrieve the MDX expression that created the subset by calling TM1ObjectPropertyGet, passing the property TM1SubsetExpression.
Security	None.
Errors	None.

Item	Description
See Also	TM1SubsetCreateEmpty TM1SubsetCreateByExpression. This is a TM1 subset property. See "Properties" for more information.

TM1SubsetCreateEmpty

Item	Description
Purpose	Creates an empty subset object.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetCreateEmpty(TM1P hPool, TM1V hDim);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hDim is a handle to the parent dimension. You can get a handle to the parent dimension by calling one of the TM1ObjectListHandle functions.
Result	Returns a handle to the subset. Subsets can be registered as public or private objects. For other IBM Cognos TM1 clients to access the new subset, you must register the subset as a public object by calling TM1ObjectRegister. To register the subset as a private object, call TM1ObjectPrivateRegister.
Security	None.
Errors	None.
See Also	TM1ObjectRegister

TM1SubsetElementDisplay

Item	Description
Purpose	Returns information necessary to draw levels, lines and plus/minus boxes corresponding to a subset element displayed in a tree hierarchy.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetElementDisplay(TM1PhPool, TM1V hSubset, TM1V iElement);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a valid handle to the subset. It is obtained either by calling TM1SubsetCreateEmpty, or by processing a dimension list property with the TM1ObjectList functions. iElement is a TM1V containing a TM1_INDEX. This value is an index into the subset corresponding to the element that you want to display.
Result	Returns a TM1V containing an encoded character string. This string contains information about the display characteristics of an element in a tree structure. You should never resolve this return value to a char *. Once you have called this function, and received the TM1V containing the string, you analyze the contents by passing the TM1V to the following functions: TM1SubsetElementDisplayLevel TM1SubsetElementDisplayTee TM1SubsetElementDisplayEll TM1SubsetElementDisplayPlus TM1SubsetElementDisplayMinus TM1SubsetElementDisplayWeight TM1SubsetElementDisplayLine
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayEll

Item	Description
Purpose	Returns a Boolean indicating if a subset element connector is an Ell (An Ell is the connector to the last element in a consolidation).
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SubsetElementDisplayEll(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the element is the last displayed element in a consolidation. In a tree structure, this element would be displayed with an ELL. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule.
Security	None.
Errors	None.

TM1SubsetElementDisplayLevel

Item	Description
Purpose	Returns a number indicating the indentation of an element in a tree structure.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1SubsetElementDisplayLevel(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	Returns a number indicating the indentation of the element in a tree display. For example, Year would return a display level of 0, while June would return a display level of 2.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayLine

Item	Description
Purpose	Returns a Boolean indicating if the connector of a subset element is a line.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SubsetElementDisplayLine(TM1U hUser, TM1V vString, TM1_INDEX Index);</pre>

Item	Description
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay. Index is a TM1_INDEX containing an integer. The integer is the position in the display tree from left to right. The first position is numbered 0. In the example below, the element May has a line in position 0 (the position corresponding to May's grandparent's display level), a tee in position 1, an icon in position 2, and the element name in position 3.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the element has a line preceding it in the position indicated by index. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayMinus

Item	Description
Purpose	Returns a Boolean indicating if the element has children displayed directly beneath it in the current element list.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SubsetElementDisplayMinus(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the element has children currently displayed directly beneath it in the subset. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayPlus

Item	Description
Purpose	Returns a Boolean indicating if a subset element has children that are not displayed directly beneath it in the current element list.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SubsetElementDisplayPlus (TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the Boolean is TRUE (1), the element has children that are currently not displayed directly beneath it in the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplaySelection

Item	Description
Purpose	Returns a Boolean indicating if the subset element is currently selected.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SubsetElementDisplaySelection (TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the element is currently selected. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayTee

Item	Description
Purpose	Returns a Boolean indicating if the connector of a subset element is a tee.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SubsetElementDisplayTee(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the Boolean is TRUE (1), the element is preceded with a tee connector in the display structure.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayWeight

Item	Description
Purpose	Returns the weight of an element.
Definition	<pre>TM1IMPORT TM1_REAL TM1API TM1SubsetElementDisplayWeight(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by TM1SystemOpen. vString is a TM1V containing a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	This function returns TM1_REAL. The number is the weight of the element in the display structure. This function applies only to elements that are currently displayed as children of a parent element.
Security	None.

Item	Description
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementListByIndexGet

Item	Description
Purpose	Returns a list of the properties of a subset by index.
Definition	TM1V vElementList = TM1SubsetElementListByIndexGet(TM1V hSubset, TM1V beginIndex, TM1V nCount)
Parameters	hSubset: a subset handle beginIndex: a start index from where the elements are to be retrieved nCount: number of elements to be retrieved
Result	vElementList: an array of elements. The array size reflects total number of elements returned, each array include element handle, name, etc. Element index in the subset matches the array (index+1) Array (): Array(SEL_HANDLE): Array(SEL_NAME): Array(SEL_TYPE): Array(SEL_INDEX): Array(SEL_DISPLAYINFO): element display info (bytes, it contains information about display level, plus/minus, selected or not, etc.) Array(SEL_LEVEL): element level in dimension Array(SEL_ALIAS): alias name, blank if subset has no alias set ...

TM1SubsetElementListByIndexGetEx

Item	Description
Purpose	Returns a list of the properties of a subset by index.

Item	Description
Definition	TM1V vElementList = TM1SubsetElementListByIndexGetEx
Parameters	hSubset: a subset handle beginIndex: a start index from where the elements are to be retrieved nCount: number of elements to be retrieved sDimName: the actual parent dimension name.
Result	vElementList: An array of elements. The array size reflects total number of elements returned, each array include element handle, name, etc. Element index in the subset matches the array (index+1). Similar to TM1SubsetElementListByIndexGet method. This API is needed for dimension editor since the subset is made from a copy of dimension off the original dimension object. Historically dimension copy does not have duplicate any attributes of original dimension. When dimension name is passed in, the server can correctly fetch all alias information for elements based on original dimension object.

TM1SubsetElementListByNamesGet

Item	Description
Purpose	Returns a list of elements in the specified subset by name.
Definition	TM1V vElementList = TM1SubsetElementListByNamesGet (TM1VhSubset, TM1V vElementNames)
Parameters	hSubset: a subset handle vElementNames: an array of element names
Result	vElementList: an array of elements. The array size reflects total number of elements returned, each array include element handle, name, etc. Element index in the subset matches the array (index+1) Array (): Array(SEL_HANDLE): Array(SEL_NAME): Array(SEL_TYPE): Array(SEL_INDEX): Array(SEL_DISPLAYINFO):NULL for this API Array(SEL_LEVEL):element level in the dimension.

TM1SubsetInsertElement

Item	Description
Purpose	Inserts an element into a subset.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetInsertElement (TM1P hPool, TM1V hSubset, TM1V hElement, TM1V iPosition);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle to the subset to which you want to add elements. hElement is a handle to the element you want to insert in the subset. Element handles are retrieved by calling the TM1ObjectList functions with the list property TM1SubsetElements(). iPosition is a TM1V containing a TM1_INDEX. This value indicates the position into which the new element is inserted in the subset. If iPosition = 0, the element is inserted at the end of the subset. Use TM1ValIndex(hPool, 0) to set it.
Result	This function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the element is successfully inserted. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. Elements can only be inserted into an unregistered subset.
Security	None. The subset into which you insert elements is unregistered. Therefore, you have exclusive access to that subset.
Errors	None.
See Also	TM1SubsetInsertSubset

TM1SubsetInsertSubset

Item	Description
Purpose	Inserts one subset into another.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetInsertSubset(TM1P hPool, TM1V hSubsetA, TM1V hSubsetB, TM1V iPosition);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubsetA is a handle of subset into which subset is to be inserted. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. hSubsetB handle of subset being inserted. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. iPosition is a TM1V containing a TM1_INDEX indicating the position to be occupied by the inserted subset. For example, if the value of the position argument is 4, the object is inserted before the fourth element of the subset. To insert an object after the last subset element, set this parameter to zero. Use TM1ValIndex(hPool, 0) to set it.
Result	This function returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the operation was successful. Use the function TM1ValBoolGet to retrieve the Boolean from the value capsule. The function inserts the elements of subset B into subset A. You can create subsets with repeated elements. For example, suppose you have two subsets like this: Subset 1 Subset 2 A A B Y C Z Inserting Subset 2 into Subset 1 with iPosition = TM1ValIndex(hPool, 0) yields a subset with the following elements: A, B, C, A, Y, Z Use TM1ValIndex(hPool, 0) to set iPosition.
Security	If the subset is a public object, you must have WRITE access to the dimension containing the subset. If the subset is unregistered or private, no security restrictions apply.
Errors	None.
See Also	TM1SubsetInsertElement

TM1SubsetListGet

Item	Description
Purpose	Returns a list of subsets for the specified dimension.

Item	Description
Definition	<pre>TM1V vSubsetList = TM1SubsetListGet(TM1V hDimension, TM1V iFlag)</pre>
Parameters	hDimension is a dimension handle iFlag:private subsets/public subsets/both (SL_GET_PRIVATESUBSET SL_GET_PUBLICSUBSET)
Result	vSubsetList is an Array of subsets for the specified dimension, each array includes subset handle, name, alias, expand above, expression(filter), levels count, elements count ... Array (): Array(SL_HANDLE): Array(SL_NAME): Array(SL_ISPUBLIC): Array(SL_ALIAS): Array(SL_IEXPANDABOVE): Array(SL_EXPRESSIONFILTER): Array(SL_LEVELSCOUNT): Array(SL_ELEMENTCOUNT): Array(SL_DIMHANDLE): Array(SL_DIMNAME): Array(SL_DIMINDEX): ...

TM1SubsetListByNamesGet

Item	Description
Purpose	Returns a list of subsets for the specified dimension by subset name.
Definition	<pre>TM1V vSubsetList = TM1SubsetListByNamesGet(TM1V hDimension, TM1v vSubsetNames, TM1V bFlag)</pre>

Item	Description
Parameters	hDimension is a dimension handle vSubsetNames is an array of subset names bFlag indicates whether to get only public subsets (default is to get private subsets first if exist, if not found, then get the public ones)
Result	vSubsetList is an Array of subsets for the specified dimension, each array includes subset handle, name, alias, expand above, expression(filter), levels count, elements count.... Array (): Array(SL_HANDLE): Array(SL_NAME): Array(SL_ISPUBLIC): Array(SL_ALIAS): Array(SL_IEXPANDABOVE): Array(SL_EXPRESSIONFILTER): Array(SL_LEVELSCOUNT): Array(SL_ELEMENTCOUNT): ...

TM1SubsetSelectByAttribute

Item	Description
Purpose	Selects elements of a subset that have an attribute matching value.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSelectByAttribute(TM1P hPool, TM1V hSubset, TM1V hAlias, TM1V sValueToMatch, TM1V bSelection);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a valid handle to the subset. It is obtained either by calling TM1SubsetCreateEmpty, or by processing a dimension list property with the TM1ObjectList functions. hAlias is a handle to an attribute. This handle is obtained by calling one of the TM1ObjectListHandle functions, and specifying the property TM1ObjectAttributes(). This call must be made on the parent dimension of the subset, not on the subset itself. sValueToMatch is a TM1V containing a string or numeric value of an attribute. bSelection is a Boolean. If the Boolean is TRUE, the element corresponding to the index is selected. If the Boolean is FALSE, the element corresponding to the element is de-selected.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. Selects elements in a subset that have a specified attribute (as indicated by hAttr) set to a specified value (as indicated by sValueToMatch).
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectByIndex

Item	Description
Purpose	Selects an element of a subset by its index.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SelectByIndex(TM1P hPool, TM1V hSubset, TM1V iPosition, TM1V bSelection);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. iPosition is an integer containing the position of the element to select. bSelection is a Boolean. If the Boolean is TRUE (1), the element corresponding to the index is selected. If the Boolean is FALSE (0), the element corresponding to the element is de-selected.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectByLevel

Item	Description
Purpose	Selects or de-selects all elements with a given level.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SelectByLevel (TM1P hPool, TM1V hSubset, TM1V iLevel, TM1V bSelection);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. iLevel is a TM1_INDEX indicating the level of element to select or de-select. bSelection is a Boolean. If this value is TRUE (1), all the elements of the specified level are selected. If the value is FALSE (0), all the elements of the given level are de-selected.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectByPattern

Item	Description
Purpose	Selects all elements whose names match a given regular expression pattern.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1SelectByPattern (TM1P hPool, TM1V hSubset, TM1V sPattern, TM1V bSelection);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. sPattern is a TM1V containing a string pattern. The pattern can contain wild card characters, such as * and ?. If the search is not for an exact match, you must use the *. For example, a search for "bird" will not find birds. A search for "bird*" will find birds. A search for "b*" will find birds. And a search for "birds" will find birds. bSelection is a TM1V containing a Boolean. If the Boolean is TRUE, elements matching the pattern are selected. If the Boolean is FALSE, elements matching the pattern are de-selected.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Elements matching the pattern are either selected or de-selected. Use the function TM1ValBoolGet to extract the Boolean. The pattern matching is applied to raw element names. Aliases applied to elements in the subset are not examined by this function.
Security	None.
Errors	None.
See Also	TM1SubsetSelectByIndex

TM1SubsetSelectionDelete

Item	Description
Purpose	Deletes selected elements from a subset.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSelectionDelete(TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. All elements that were previously selected through one or more of the TM1SubsetSelect functions are now deleted from the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectionInsertChildren

Item	Description
Purpose	Takes each selected element and inserts its children, if any, directly under the element in the list. This function is used to drill down on the elements in a subset.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSelectionInsertChildren (TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset into which you want to insert elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The subset referenced by the handle now contains the children of the elements that were previously selected in the subset. If the children are already present, this function inserts them again. It is the application's responsibility to check for and eliminate duplicates.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions. TM1SubsetSelectionInsertParents

TM1SubsetSelectionInsertParents

Item	Description
Purpose	Inserts the parents of each selected element directly above the element in the list.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSelectionInsertParents (TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset into which you want to insert elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	Inserts the parents of each selected element above the element. If an element is a member of more than one consolidation, all of its parents are inserted into the list. If the parents are already present, this function inserts them again. It is the application's responsibility to check for and eliminate duplicates.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions. TM1SubsetSelectionInsertChildren

TM1SubsetSelectionKeep

Item	Description
Purpose	Removes all elements from the subset that are not selected.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSelectionKeep(TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle to the subset. It is obtained with TM1SubsetCreateEmpty.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The function removes all elements from a subset that are not selected by one of the TM1SubsetSelect functions.
Security	None.

Item	Description
Errors	None.
See Also	TM1SubsetCreateEmpty

TM1SubsetSelectNone

Item	Description
Purpose	Clears the selection flag from any selected elements in a subset.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSelectNone(TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset into which you want to insert elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	Clears the selection flag for all elements in the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSort

Item	Description
Purpose	Sorts the elements in a subset alphabetically.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSort(TM1P hPool, TM1V hSubset, TM1V bSortDown);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubset is a handle of the subset into which you want to insert elements. bSortDown is a TM1V containing a Boolean. If the Boolean is FALSE (0), the elements in the subset are sorted in alphabetical order from A to Z. If the Boolean is TRUE (1), the subset elements are sorted in reverse alphabetical order from Z to A.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the subset elements are sorted from Z through A. If the Boolean is 0, the subset elements are sorted from A through Z. Use the parameter bSortDown to set the sorting order. The sorting is applied to raw element names. This function does not examine aliases applied to elements in the subset.
Security	None.
Errors	None.
See Also	Other TM1Subset functions.

TM1SubsetSortByHierarchy

Item	Description
Purpose	Sorts the elements of a subset according to their parent / child relationships.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSortByHierarchy(TM1P hPool, TM1V hSubset);</pre>
Parameters	hPool is an input parameter. This is an IBM Cognos TM1 pool handle as returned by the function TM1ValPoolCreate. hSubset is an input parameter. This is a handle to the subset that you want to sort.
Result	This function sorts the subset as follows: All the elements that have neither parents nor children are grouped first, in alphabetical order. All the consolidated elements containing at least one child element are sorted in alphabetical order. All the child objects are grouped below their parents, and are sorted in alphabetical order.
Security	None.
Errors	None.
See also	TM1SubsetSort

TM1SubsetSubtract

Item	Description
Purpose	Removes a set of elements from a subset.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetSubtract(TM1P hPool, TM1V hSubsetA, TM1V hSubsetB);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hSubsetA is a handle of the subset from which you want to delete elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. hSubsetB is a handle of the subset whose member elements you want to delete from Subset A. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function eliminates from Subset A any elements that are common to both Subset A and Subset B.
Security	None.
Errors	None.
See Also	TM1SubsetInsertSubset

TM1SubsetUpdate

Item	Description
Purpose	Replaces a registered subset with a new one.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SubsetUpdate(TM1P hPool, TM1V hOldSubset, TM1V hNewSubset);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hOldSubset is a handle of the registered subset to be replaced. hNewSubset is a handle to the subset that replaces the old one.

Item	Description
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The old subset is destroyed and replaced with the new one. All affected views are updated accordingly.
Security	The client must have ADMIN rights to the dimension being updated.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Subset functions.

TM1SystemAdminHostGet

Item	Description
Purpose	Retrieves the name of the IBM Cognos TM1 admin host server.
Definition	<pre>TM1IMPORT CHAR * TM1API TM1SystemAdminHostGet(TM1U hUser);</pre>
Parameters	hUser is a long. It is a valid user handle obtained with TM1SystemOpen.
Result	This function returns a string. The string is the name of the TM1 admin host.
Security	None.
Errors	None.
See Also	TM1SystemAdminHostSet

TM1SystemAdminHostSet

Item	Description
Purpose	Sets the name of the AdminHost server.
Definition	<pre>TM1IMPORT void TM1API TM1SystemAdminHostSet(TM1U hUser, char * szAdminHosts);</pre>

Item	Description
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. szAdminHosts is a NULL-terminated string specifying a list of host names separated by commas.
Result	This function must be called before any TM1SystemServer functions, and may be called at any time to reset the list of available servers. This function does not affect existing connections.
Security	None.
Errors	None.
See Also	TM1SystemOpen

TM1SystemBuildNumber

Item	Description
Purpose	Returns a string corresponding to the build number of the IBM Cognos TM1 server.
Definition	TM1IMPORT CHAR * TM1API TM1SystemBuildNumber(void)
Parameters	None.
Result	The function returns a build string in the format: major.minor.patchrev.intbuild {special version} for example 9.5.20100.1234 - Hotfix patchrev refers to patch and revision. For example, TM1 9.5 MR2 FP1 build 1234. The remaining portion is an optional string that is optional.
Security	None.
Errors	None.
See Also	TM1SystemOpen

TM1SystemClose

Item	Description
Purpose	Disconnects the user from the API and releases resources.

Item	Description
Definition	TM1IMPORT void TM1API TM1SystemClose(TM1U hUser);
Parameters	hUser is a valid user handle obtained with TM1SystemOpen.
Results	Before you can disconnect from the server, you must run TM1SystemServerDisconnect(). Then, when you run TM1SystemClose(), the user is disconnected from the API and all resources are released. Any existing connections are closed.
Security	None.
Errors	None.
See Also	TM1SystemOpen

TM1SystemGetServerConfig

Item	Description
Purpose	Returns the server configuration settings.
Definition	TM1P hPool, TM1V vServerName
Parameters	hpool: a cube handle vServerName: name of server
Result	Array will have the following format: 0 - Security mode 1 - Array in the following format depending on the security mode: Mode: MIXED_SECURITY_MODE or INTEGRATED_SECURITY_MODE [0] Principal Name (string) [1] Package Name Mode: CAM_SECURITY_MODE [0] CAM URI (string) [1] CAM Web URI (string) [2] Client Ping CAM Passport Interval [3] Server CAM Dispatcher URI (string)

TM1SystemOpen

Item	Description
Purpose	Connects the user to the API.
Definition	<pre>TM1IMPORT TM1U TM1API TM1SystemOpen(void);</pre>
Parameters	None.
Result	The function returns a user handle. Typically, this user handle is used to create a pool handle with TM1ValPoolCreate. The pool handle is then passed to other API calls as an argument. This function is part of the API initialization sequence required by every IBM Cognos TM1 API program. See "Connecting to the API" for more information.
Security	None.
Errors	None.
See Also	TM1SystemClose

TM1SystemProgressHookSet

Item	Description
Purpose	Sets up a callback procedure to handle progress messages.
Definition	<pre>TM1IMPORT void TM1API TM1SystemProgressHookSet(TM1UhUser, TM1_HOOK pHook);</pre>
Parameters	hUser is a user handle returned by TM1SystemOpen. pHook is a pointer to a callback function. The callback function is declared as follows: <pre>void CALLBACK ProgressFunction(unsigned char message, unsigned char action, unsigned long param, char * name);</pre> Where message is one of the following constants: TM1SystemProgressMessageOpening This message is sent when starting a process that may take a long time to complete. TM1SystemProgressMessageRunning This message is sent every second while the process is running. TM1SystemProgressMessageClosing This message is sent when the process is completed.

Item	Description
	action is a constant that specifies the kind of progress that is taking place. The TM1ProgressAction constants are defined in TM1API.h. param is an unsigned long indicating the progress. Param is one of these constants: TM1SystemProgressTypePercent; The progress value will be a number between 0 and 100 indicating a percent completion. TM1SystemProgressTypeCounter; The progress value will indicate the number of steps completed. name provides the name of the object being processed. (Opening message only). If pHook is a NULL pointer, TM1SystemProgressHookSet clears a previously-set callback function.

TM1SystemServerClientName

Item	Description
Purpose	Returns a client's name.
Definition	<pre>TM1IMPORT char * TM1API TM1SystemServerClientName (TM1U hUser, unsigned index);</pre>
Parameters	hUser is an IBM Cognos TM1 user handle returned by the function TM1SystemServerConnect. index is an integer. This integer is an offset into the list of available servers currently available to the client. These servers are listed in the admin server user interface.
Result	The function returns a string. The string contains the name of the current user. Use TM1ValStringGet to retrieve information.
Security	None.
Errors	None.
See Also	TM1SystemServerConnect

TM1SystemServerConnect

Item	Description
Purpose	Connects a client to a server. Use this function to start an IBM Cognos TM1 session if your TM1 server is configured for standard TM1 authentication or LDAP authentication. Use TM1SystemServerConnectIntegratedLogin if it is configured for Integrated Login.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SystemServerConnect(TM1P hPool, TM1V sServerName, TM1V sClientName, TM1V sPassword);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. sServerName is a string value containing the name of the server. sClientName is a string value containing the name of the client. sPassword is a string value containing the password in plain or encrypted form.
Result	The function returns a handle to the server. This function is part of the API initialization sequence required by every IBM Cognos TM1 API program. See the section "Connecting to the API" for more information.
Errors	<pre>TM1ErrorSystemServerNotFound TM1ErrorSystemServerConnectionFailed TM1ErrorSystemServer ClientAlreadyConnected TM1ErrorSystemServerClientNotFound TM1ErrorSystemServerClientPassword Invalid TM1ErrorSystemServerClientPassword Expired TM1ErrorSystemServerClient ExceedMaxLogonNumber TM1ErrorClientMaximumPortsExceeded</pre>
See Also	<pre>TM1SystemOpen TM1SystemServerDisconnect</pre>

TM1SystemServerConnectIntegratedLogin

Item	Description
Purpose	Connects a client to a server using integrated login. Integrated login allows you to use your Microsoft Windows security system to authenticate IBM Cognos TM1 users. For more information on setting up integrated login, see the IBM Cognos TM1 <i>Operations Guide</i>. Use the function TM1SystemServerConnect instead of this function if your TM1 server is configured for standard TM1 authentication or LDAP authentication.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SystemServerConnect IntegratedLogin(TM1P hPool, TM1V sServerName);</pre>
Parameters	hPool is a long. This TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. sServerName is a long. This TM1 value capsule contains a string value containing the name of the server.
Result	The function returns a handle to the server. This function attempts to connect to the server through integrated login. The login is attempted using the Microsoft Windows domain name under which the user logged in to the network. for example, suppose you follow this sequence: You log in to a Microsoft Windows workstation as Stewart in the Germany domain. You run an application that includes TM1SystemServerConnectIntegratedLogin. The API will try to log in to IBM Cognos TM1 using the id Stewart. The TM1 server must be configured to accept integrated logins. (The IntegratedSecurityMode parameter in tm1s.cfg must be set to 2 or 3.)
Errors	<pre>TM1ErrorSystemServer IntegratedSecurityRefused TM1ErrorSystemServer IntegratedSecurityRequired</pre>
See Also	<pre>TM1SystemOpen TM1SystemServerConnect TM1SystemServerDisconnect</pre>

TM1SystemServerDisconnect

Item	Description
Purpose	Disconnects a user from a server.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1SystemServerDisconnect(TM1P hPool, TM1V hServer);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle -- a TM1V object value -- to the connected server.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	<pre>TM1ErrorSystemServer ClientNotConnected</pre>
See Also	Other TM1SystemServer functions.

TM1SystemServerHandle

Item	Description
Purpose	Returns the handle to a server given its name. This function can only returns a handle for a server to which you have already established a connection.
Definition	<pre>TM1IMPORT TM1V TM1API TM1SystemServerHandle(TM1U hUser, char * szName);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. szName is a string value containing the name of the server.
Result	The function returns a TM1_OBJECT which is the handle to the requested server.
Security	None.
Errors	If the function fails, it returns a TM1V containing a TM1_BOOL with a value of 0. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
See Also	Other TM1SystemServer functions.

TM1SystemServerName

Item	Description
Purpose	Returns the name of a server in the list of available servers given an index.
Definition	<pre>TM1IMPORT char * TM1API TM1SystemServerName(TM1U hUser, unsigned index);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. index is an unsigned integer, starting from 1. It indicates the position of the server in the AdminHost list.
Result	The function returns the name of the server at the specified position in the AdminHost list. The function returns a null string if the index is out of range.
Security	None.
Errors	None.
See Also	Other TM1SystemServer functions.

TM1SystemServerNof

Item	Description
Purpose	Returns the number of available servers.
Definition	<pre>TM1IMPORT int TM1API TM1SystemServerNof(TM1U hUser);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen.
Result	The function returns the number of available servers. In order to receive the correct number of servers, you must call TM1SystemServerReload before you call this function.
Security	None.
Errors	None.
See Also	Other TM1SystemServer functions.

TM1SystemServerReload

Item	Description
Purpose	Loads information from the IBM Cognos TM1 Admin Server into the API.
Definition	<pre>TM1IMPORT void TM1API TM1SystemServerReload(TM1U hUser);</pre>
Parameters	hUser is a long. It is a valid user handle obtained with TM1SystemOpen.
Result	The function loads information from the TM1 Admin Server into the API. In order to get an accurate count of available servers, you must call TM1SystemServerReload, then call TM1SystemServerNof.
Security	None.
Errors	None.
See Also	TM1SystemServerNof

TM1SystemServerStart

Item	Description
Purpose	Starts the IBM Cognos TM1 server.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SystemServerStart(TM1U hUser, char * szName, char * szDataDirectory, char * szAdminHost, char * szProtocol, int iPortNumber);</pre>
Parameters	hUser is a user handle as returned by TM1SystemOpen. szName is a null-terminated string containing the name of the server to start. szDataDirectory is a null-terminated string containing the path of the TM1 data directory. szAdminHost is a null-terminated string containing the path of the TM1 admin directory. szProtocol is a null-terminated string containing the protocol to use to connect to the server. For TCP/IP, you should use the string "tcp." For IPX, you should use the string "ipx." iPortNumber is an integer containing the port number of the server. The default port number for the TM1 server is 5000.

Item	Description
Result	The function returns a TM1_BOOL. If the Boolean is 1, the operation starts an IBM Cognos TM1 server. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function allows you to start a server on the local machine only, not on other machines in the network. This function does not work on a TM1 server running on UNIX.
Security	None.
Errors	None.
See Also	TM1SystemServerStop

TM1SystemServerStartEx

Item	Description
Purpose	Starts the IBM Cognos TM1 server using a configuration file. SSL configuration parameters are specified in the server configuration file. The TM1ServerStart C-API does not take a parameter for specifying a configuration file and therefore can not start the server using non-default SSL parameters. To specify a configuration file when starting a server, use the TM1SystemServerStartEx C-API. This API allows the entire command line to be specified and therefore allows the use of a -z parameter to specify a configuration file.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SystemServerStartEX(TM1U hUser, char * szName, char * szDataDirectory, char * szAdminHost, char * szProtocol, int iPortNumber);</pre>
Parameters	hUser is a user handle as returned by TM1SystemOpen. szName is a null-terminated string containing the name of the server to start. szDataDirectory is a null-terminated string containing the path of the IBM Cognos TM1 data directory. szAdminHost is a null-terminated string containing the path of the TM1 admin directory. szProtocol is a null-terminated string containing the protocol to use to connect to the server. For TCP/IP, you should use the string "tcp." For IPX, you should use the string "ipx." iPortNumber is an integer containing the port number of the server. The default port number for the TM1 server is 5000.

Item	Description
Result	The function returns a TM1_BOOL. If the Boolean is 1, the operation starts a TM1 server. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function allows you to start a server on the local machine only, not on other machines in the network. This function does not work on a TM1 server running on UNIX.
Security	None.
Errors	None.
See Also	TM1SystemServerStop

TM1SystemServerStop

Item	Description
Purpose	Stops an IBM Cognos TM1 server.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1SystemServerStop(TM1U hUser, char * szName, TM1_BOOL bSave);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. szName is a null-terminated string containing the name of the server to stop. bSave is a Boolean. If the Boolean is TRUE (1), changes to TM1 server data in memory are written to the hard disk before the server is brought down. If the Boolean is FALSE (0), no changes are written to disk before the server is brought down.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the function stops the execution of a local TM1 server executable. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.
See Also	TM1SystemServerStart

TM1SystemVersionGet

Item	Description
Purpose	Returns the current version of the API.
Definition	<pre>TM1IMPORT int TM1API TM1SystemVersionGet(void);</pre>
Result	Returns an integer indicating the version of the API multiplied by 100. For example version 2.5 will result in 250.
Security	None.
Errors	None.

TM1UserKill

Item	Description
Purpose	Creates a new worker thread to authenticate the thread, cancels the client and logs the user out as a result of the user pressing a cancel button. This API completely stops, on behalf of the current client, the current client operation being executed on the server.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1UserKill(TM1U hUser, TM1V hServer)</pre>
Parameters	<i>hUser</i> is a handle to the current user. <i>hServer</i> is a handle to the current server. TM1UserKill will authenticate the cancel request, cancel the current transaction, and log the user out.
Result	None.
Security	None.
Errors	Upon completion, a success or failure is returned by the server.
See Also	TM1CancelClientJob

TM1ValArray

Item	Description
Purpose	Constructs an array value capsule.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValArray(TM1P hPool, TM1V * InitArray, TM1_INDEX MaxSize);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. InitArray is an array of value handles with which the array will be initialized. MaxSize is the maximum number of values that the array can hold.
Result	The function returns the handle to the array value capsule created. The array has no values. You must add values to the array with the function TM1ValArraySet. If the value cannot be created, perhaps because of lack of memory, the function returns a TM1V containing a TM1_BOOL. If the Boolean is 0, the operation was unsuccessful. Use the function TM1ValBoolGet to extract the Boolean.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValArrayGet

Item	Description
Purpose	Retrieves a component of an array value.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValArrayGet(TM1U hUser, TM1V vArray, TM1_INDEX Index);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vArray is a handle to an array value. Index is a one-based position within the array.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. The function returns the value handle stored at the position given by Index.
Security	None.

Item	Description
Errors	The function returns a zero if errors are encountered.
See Also	Other TM1ValArray functions.

TM1ValArrayMaxSize

Item	Description
Purpose	Returns the number of elements in an array.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValArrayMaxSize(TM1U hUser, TM1V vArray);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vArray is an IBM Cognos TM1 value capsule containing an array value.
Result	This function returns the number of values in a TM1 array value capsule. The function returns zero if an error occurs.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValArraySet

Item	Description
Purpose	Updates a component of an array value.
Definition	<pre>TM1IMPORT void TM1API TM1ValArraySet(TM1V vArray, TM1V vValue, TM1_INDEX Index);</pre>
Parameters	vArray is a handle to an array value. vValue is the value handle to be stored in the array. Index is a (one-based) position within the array.

Item	Description
Result	The current value handle at position Index within array vValue is replaced by newval. Note that overwriting an object handle in an array does not destroy the underlying object on the IBM Cognos TM1 server.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValArraySetSize

Item	Description
Purpose	Establishes an array value of a given size.
Definition	<pre>TM1IMPORT void TM1API TM1ValArraySetSize(TM1V vArray, TM1_INDEX Index);</pre>
Parameters	vArray is a handle to an array value. Index is a index indicating the size of the array you are creating.
Result	Establishes the value capsule as an array of Index elements. Call TM1ValArray before you call this function.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValBool

Item	Description
Purpose	Constructs a Boolean value capsule.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValBool(TM1P hPool, TM1_BOOL InitBool);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. InitBool is the value to be stored in the capsule.
Result	The function returns the handle to the value capsule created.

Item	Description
Security	None.
Errors	If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
See Also	Other TM1ValBool functions.

TM1ValBoolGet

Item	Description
Purpose	Retrieves the contents of a Boolean value capsule.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1ValBoolGet(TM1U hUser, TM1V vBool);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vBool is a handle to the value capsule containing a Boolean.
Result	The function returns the Boolean contents of the value capsule.
Security	None.
Errors	If there is an error, the function returns zero.
See Also	Other TM1ValBool functions.

TM1ValBoolSet

Item	Description
Purpose	Update the contents of a Boolean value capsule.
Definition	<pre>TM1IMPORT void TM1API TM1ValBoolSet(TM1V vBool, TM1_BOOL bool);</pre>
Parameters	vBool is a handle to the Boolean value capsule whose contents is to be updated. bool is the value used to update the capsule.
Result	The function updates the target Boolean value capsule with the new value.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1ValBool functions.

TM1ValErrorCode

Item	Description
Purpose	Extracts the error code from an error value.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValErrorCode(TM1U hUser, TM1V vError);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vError is a handle to the error value capsule whose contents is to be retrieved.
Result	The function returns the error code. You can pass this error code to TM1ValErrorString to receive an error message string.
Security	None.
Errors	If an error occurs, the function returns zero.
See Also	TM1ValErrorString

TM1ValErrorString

Item	Description
Purpose	Retrieves a string corresponding to an IBM Cognos TM1 error code.
Definition	<pre>TM1IMPORT LPSTR TM1API TM1ValErrorString(TM1U hUser, TM1V vValue);</pre>
Parameters	hUser is a user handle as returned by TM1SystemOpen. vValue is a TM1V containing a TM1 error code.
Result	Returns a string corresponding to a TM1 error code.
Security	None.
Errors	None.
See Also	TM1ValErrorCode

TM1ValIndex

Item	Description
Purpose	To construct a value capsule containing an index (32-bit integer).
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValIndex(TM1P hPool, TM1_INDEX InitIndex);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. InitIndex is the value to be stored in the capsule.
Result	The function returns the handle to the value capsule created.
Security	None.
Errors	If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
See Also	TM1ValIndexGet TM1ValIndexSet

TM1ValIndexGet

Item	Description
Purpose	Retrieves the contents of an index value capsule.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValIndexGet(TM1U hUser, TM1V vIndex);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vIndex is a handle to the value capsule whose contents is to be retrieved.
Result	The function returns a TM1V containing a TM1_INDEX. If the function returns zero, the operation was not successful. The function returns the index contents of the value capsule.
Security	None.
Errors	None.
See Also	Other TM1ValIndex functions.

TM1ValIndexSet

Item	Description
Purpose	Update the contents of an index value capsule.
Definition	<pre>TM1IMPORT void TM1API TM1ValIndexSet(TM1V vIndex, TM1_INDEX index);</pre>
Parameters	vIndex is the value capsule whose contents is to be updated. index is the value used to update the capsule.
Result	The function updates the target index value capsule with the new value.
Security	None.
Errors	None.
See Also	Other TM1ValIndex functions.

TM1ValsUndefined

Item	Description
Purpose	Tests whether a value is of type TM1CubeCellValueUndefined().
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1ValIsUndefined(TM1U hUser, TM1V Value);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. Value is a handle to the value to be tested.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the value is of type TM1CubeCellValueUndefined. Otherwise, the function returns zero. Use the function TM1ValBoolGet to extract the Boolean.
Security	None.
Errors	None.

TM1ValsChanged

Item	Description
Purpose	Tests whether a value has been changed inside the sandbox.

Item	Description
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1ValIsUpdatable(TM1U hUser, TM1V Value);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. Value is a handle to the value to be tested.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the client can update the value. Otherwise, it returns a 0. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function applies to cell values and object properties. This function has no meaning when cell values are from base cube.
Security	None.
Errors	None.
See Also	TM1ValIsUndefined

TM1ValIsUpdatable

Item	Description
Purpose	Tests whether a value retrieved from a server can be updated.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1ValIsUpdatable(TM1U hUser, TM1V Value);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. Value is a handle to the value to be tested.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the client can update the value. Otherwise, it returns a 0. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. This function applies to cell values and object properties.
Security	None.
Errors	None.
See Also	TM1ValIsUndefined

TM1ValObject

Item	Description
Purpose	To construct a value capsule containing an object handle.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValObject(TM1P hPool, TM1_OBJECT * InitObject);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. InitObject points to the object handle to be stored in the capsule.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectCanRead

Item	Description
Purpose	Determines whether the client has READ access to an object.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1ValObjectCanRead(TM1U hUser, TM1V vObject);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vObject is a handle to the value containing the object handle.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, one of the groups to which the client belongs has READ or higher rights to the object. Otherwise, it returns zero. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectCanWrite

Item	Description
Purpose	Determines whether the client has WRITE access to an object.
Definition	<pre>TM1IMPORT TM1_BOOL TM1API TM1ValObjectCanWrite(TM1U hUser, TM1V vObject);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vObject is a handle to the value containing the object handle.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, one of the groups to which the client belongs has WRITE or higher rights to the object, provided that the object is not reserved or locked. Otherwise, it returns zero. Use the function TM1ValBoolGet to extract the Boolean. If there is an error, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectGet

Item	Description
Purpose	Retrieves the contents of an object value capsule.
Definition	<pre>TM1IMPORT void TM1API TM1ValObjectGet(TM1U hUser, TM1V vObject, TM1_OBJECT * pObject);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vObject is a handle to the value capsule whose contents is to be retrieved. pObject points to the area to receive the extracted object handle.
Result	This function returns nothing when successful. pObject contains the TM1_OBJECT that was extracted from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectSet

Item	Description
Purpose	Update the contents of an object value capsule.
Definition	<pre>TM1IMPORT void TM1API TM1ValObjectSet(TM1V vObject, TM1_OBJECT * pObject);</pre>
Parameters	vObject is a TM1V containing the object handle that is to be updated. pObject is a pointer to the object. This object is used to update the capsule.
Result	The function updates the target object value capsule with the new value.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectType

Item	Description
Purpose	Retrieves the type of object.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValObjectType (TM1U hUser, TM1V vObject);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vObject is a value capsule containing an IBM Cognos TM1 object handle.
Result	Returns an integer. The integer is one of the TM1Type constants defined in Tm1api.h. For example, if the object is a cube, TM1ObjectType returns the constant Tm1TypeCube().
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValPoolCount

Item	Description
Purpose	Returns the number of values stored in a value pool.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValPoolCount(TM1P hPool);</pre>
Parameters	hPool is a handle to a pool of values.
Result	The function returns a TM1_INDEX. The value indicates the number of values in the pool. If the function returns zero, the operation was not successful.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolCreate

Item	Description
Purpose	Creates a new value pool.
Definition	<pre>TM1IMPORT TM1P TM1API TM1ValPoolCreate(TM1U hUser);</pre>
Parameters	hUser is a user handle obtained with TM1SystemOpen.
Result	The function returns a handle to the pool. The handle is valid until the TM1ValPoolDestroy function is called. If there is an error, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolDestroy

Item	Description
Purpose	Clears a value pool.

Item	Description
Definition	<pre>TM1IMPORT void TM1API TM1ValPoolDestroy(TM1P hPool);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate.
Result	This function does not return a value. The value pool is cleared. Any value handles referring to the value pool become invalid. Using such handles will cause unpredictable results. The memory occupied by the value pool is retained by the IBM Cognos TM1 server. It is not released back to the operating system.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolGet

Item	Description
Purpose	Retrieves a value from a value pool.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValPoolGet(TM1P hPool, TM1_INDEX Index);</pre>
Parameters	hPool is a handle to a pool of values. Index specifies the relative position (zero-based) of the value within the pool.
Result	Returns a TM1V containing a handle. This value is the value handle at the position given by Index in the value pool. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolMemory

Item	Description
Purpose	Retrieves the amount of memory in bytes currently used by a value pool.
Definition	<pre>TM1IMPORT unsigned long TM1API TM1ValMemory(TM1P hPool);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate.
Result	This function returns an unsigned long containing the size of the value pool. The initial size of the value pool is 1 kilobyte.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValReal

Item	Description
Purpose	To construct a value capsule containing a real value.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValReal(TM1P hPool, TM1_REAL InitReal);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. InitReal is the value to be stored in the capsule.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ValRealGet

Item	Description
Purpose	Retrieves the contents of a real value capsule.
Definition	<pre>TM1IMPORT TM1_REAL TM1API TM1ValRealGet(TM1U hUser, TM1V vReal);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vReal is a TM1V containing a real.
Result	The function returns the real contents of the value capsule. If there is an error, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ValRealSet

Item	Description
Purpose	Update the contents of a real value capsule.
Definition	<pre>TM1IMPORT void TM1API TM1ValRealSet(TM1V vReal, TM1_REAL Real);</pre>
Parameters	vReal is a TM1V whose contents is to be updated. Real is the value used to update the capsule.
Result	This function does not return a value. The value of Real is increased into vReal.
Security	None.
Errors	None
See Also	Other TM1Val functions.

TM1ValString

Item	Description
Purpose	Constructs a value capsule containing a string.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValString(TM1P hPool, char * szString, TM1_INDEX Maxsize);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. szString is the value to be stored in the capsule. Maxsize is an integer indicating the maximum length of a string than can be held in this value capsule. A value of zero means that the maximum length should be equal to the length of the string being passed in.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringEncrypt

Item	Description
Purpose	Constructs a value capsule containing an encrypted string.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValStringEncrypt (TM1P hPool, char * szString, TM1_INDEX Maxsize);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. szString is the value to be stored in the capsule. Maxsize is an integer indicating the maximum length of a string than can be held in this value capsule. A value of zero means that the maximum length should be equal to the length of the string being passed in.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1ValString functions.

TM1ValStringGet

Item	Description
Purpose	The amount of memory (in bytes) allocated to hold the string.
Definition	<pre>TM1IMPORT char * TM1API TM1ValStringGet(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vString is a TM1V containing a string value.
Result	The function returns a pointer to the string. The end of the string is marked by a zero byte. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringMaxSize

Item	Description
Purpose	The amount of memory (in bytes) allocated to hold the Unicode string.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValStringMaxSize(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vString is a TM1V containing a string value.
Result	The function returns the size of the longest string that can be saved in the string value capsule. If there is an error, the function returns a zero.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1ValString functions.

TM1ValStringWMaxSize

Item	Description
Purpose	Returns the maximum string size that can be stored in a string capsule.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValStringWMaxSize(TM1UhUser, TM1V vString);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vString is a TM1V containing a string value.
Result	The function returns the size of the longest string that can be saved in the string value capsule. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringSet

Item	Description
Purpose	Set a string object to a passed "char *" based string pointer. The string is assumed to be in the local character encoding of the machine.
Definition	<pre>TM1IMPORT void TM1API TM1ValStringSet(TM1V vString, char * String);</pre>
Parameters	vString is a TM1V whose contents is to be updated. String is a pointer to a string that is used to update the capsule. The length of the new string value should not exceed the maximum length of the string specified when the capsule was originally created. If it does exceed this length, the new value is truncated accordingly.
Result	This function does not return a value. The value of String is inserted into the vString variable.

Item	Description
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringSetW

Item	Description
Purpose	Set a string object to a passed Unicode string pointer.
Definition	<pre>TM1IMPORT void TM1API TM1ValStringSetW(TM1V vString, TM1_UTF16_T *utf16String);</pre>
Parameters	vString is a TM1V whose contents is to be updated. String is a pointer to a string that is used to update the capsule. The length of the new string value should not exceed the maximum length of the string specified when the capsule was originally created. If it does exceed this length, the new value is truncated accordingly.
Result	This function does not return a value. The value of String is inserted into the vString variable.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringSetUTF8

Item	Description
Purpose	Set a string object to a passed UTF-8 encoded Unicode string.
Definition	<pre>TM1IMPORT void TM1API TM1ValStringSetUTF8(TM1V vString, TM1_UTF8_T *utf16String);</pre>

Item	Description
Parameters	vString is a TM1V whose contents is to be updated. String is a pointer to a string that is used to update the capsule. The length of the new string value should not exceed the maximum length of the string specified when the capsule was originally created. If it does exceed this length, the new value is truncated accordingly.
Result	This function does not return a value. The value of String is inserted into the vString variable.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringUTF8MaxSize

Item	Description
Purpose	The amount of memory (in bytes) allocated to hold the UTF-8 encoded Unicode string.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValStringUTF8MaxSize(TM1U hUser, TM1V vString);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vString is a TM1V containing a string value.
Result	The function returns the size of the longest string that can be saved in the string value capsule. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValType

Item	Description
Purpose	Returns the type of an object. This is depreciated, one should use TM1ValTypeEx() otherwise all the types below are returned as "TM1ValTypeString()". TM1ValTypeEx - returns the type of an object. This will return different string types: TM1ValTypeStringW() – a UTF-8 encoded Unicode string. TM1ValTypeString() – A string encoded in the local encoding of the machine. TM1ValTypeBinary() – The object contains a binary object as a stream of bytes.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValType(TM1U hUser, TM1V vValue);</pre>
Parameters	hUser is a valid user handle obtained with TM1SystemOpen. vValue is a handle to the value capsule whose type is to be retrieved.
Result	The function returns one of the following constants: TM1ValTypeReal(); TM1ValTypeString(); TM1ValTypeIndex(); TM1ValTypeBool(); TM1ValTypeObject(); TM1ValTypeError(); TM1ValTypeArray(); For example, if the value capsule returns TM1ValTypeReal(), the data in the value capsule is TM1_REAL. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ValTypeEx

Item	Description
Purpose	Retrieves a duplicate type of a value.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValTypeEx(TM1U hUser, TM1V Value);</pre>
Parameters	
Result	
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ValTypeIsString

Item	Description
Purpose	Returns true if the passed object is a string.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValTypeIsString(TM1U hUser, TM1V Value);</pre>

TM1ValTypeIsBinary

Item	Description
Purpose	Returns true if the passed object is a binary object, which is a stream of bytes.
Definition	<pre>TM1IMPORT TM1_INDEX TM1API TM1ValTypeIsBinary(TM1U hUser, TM1V Value);</pre>

TM1ViewArrayColumnsNof

Item	Description
Purpose	Returns the number of columns in the view array including columns for dimensions and data.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayColumnsNof(TM1P hPool, TM1V hView);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube property TM1CubeViews.
Result	Returns the number of columns in a view.
Security	None.
Errors	None.
See Also	TM1ViewArrayRowsNof

TM1ViewArrayConstruct

Item	Description
Purpose	Constructs a two dimensional array of data that can be used to display the data of a view. When you use TM1ViewArrayConstruct to access a registered view (via the hView parameter), IBM Cognos TM1 applies a server lock while the view is calculated.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayConstruct (TM1P hPool, TM1V hView);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the values in the view are available for retrieval. To extract values from a view, you would typically call functions in this order: TM1ViewArrayConstruct TM1ViewArrayRowsNof TM1ViewArrayColumnsNof TM1ViewArrayValueGet
Security	None.
Errors	None.

Item	Description
See Also	TM1ViewArrayValueGet TM1ViewArrayRowsNof TM1ViewArrayColumnsNof

TM1ViewArrayDestroy

Item	Description
Purpose	Destroys view array constructed by TM1ViewArrayConstruct.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayDestroy(TM1P hPool, TM1V hView);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
Result	Destroys a view array created with the function TM1ViewArrayCreate.
Security	None.
Errors	None.
See Also	TM1ViewCreate

TM1ViewArrayRowsNof

Item	Description
Purpose	Returns the number of rows in the view including rows for dimensions and data.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayRowsNof(TM1P hPool, TM1V hView);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube property TM1CubeViews.
Result	Returns the number of rows in a view array.
Security	None.

Item	Description
Errors	None.
See Also	TM1ViewArrayColumnsNof

TM1ViewArrayValueByRangeGet

Item	Description
Purpose	Retrieves a value from a view using a range.
Definition	TM1V hView, TM1V iRowStart, TM1V iColStart, TM1V iRowEnd, TM1V iColEnd
Parameters	hServer: a view handle iRowStart: the start row index at which cell values are to be retrieved, 0-based index iColStart: the start column index at which cell values are to be retrieved, 0-based index iRowEnd: the end row index iColEnd: the end column index
Result	vCellData: raw value of view cells from the view array. It is similar to Array(GV_CELLDATA) in TM1GetViewByName function. Array(index): Array of cells for a given row number offset the iRowStart Array(iColNum): raw value of a cell at a given column number of the row offset the iColStart

TM1ViewArrayValueGet

Item	Description
Purpose	Retrieves a single value from a view.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayValueGet (TM1P hPool, TM1V hView, TM1V iColumn, TM1V iRow);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube property TM1CubeViews. iColumn is a TM1V containing an integer. The integer is a 1-based number corresponding to the column of the value you want to retrieve. iRow is a TM1V containing an integer. The integer is a 1-based number corresponding to the row of the value you want to retrieve.
Result	Returns a single cell value from a view. This value can be any one of the following: number value (either an integer or a real number) a string value an index into a subset. This is the case when you retrieve values in either the first row or first column of the view array. You can use the index to retrieve the handle of the subset element within the row or column subset. To do this, call the function TM1ObjectListHandleByIndexGet, passing the index from the view array, the subset handle, and the property TM1SubsetElements(). NULL. You must construct a view array by calling TM1ViewArrayConstruct before you can successfully call this function.
Security	None.
Errors	None.
See Also	TM1ViewArrayConstruct

TM1ViewArrayValuePickListGet

Item	Description
Purpose	Call this function to get the PickList for a cell in a view.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayValuePickListGet(TM1P hPool, TM1V hView, TM1V iColumn, TM1V iRow)</pre>

Item	Description
Parameters	hView - view that contains the cell you wish to get the PickList for iColumn - the cell's column in the view iRow - the cell's row in the view
Result	A two-element array. The first element is an enumeration indicating what type of PickList is contained in the second element. If this enumeration is 'TYPE_SUBSET' then the second element of the array is also an array with two elements. The first element is a string representing the subset's dimension's name and the second element is a string representing the subset's name. If the type enumeration is 'TYPE_STATIC' then the second element of the array is an array of strings with each string being an element of the PickList. If the type enumeration is 'TYPE_DIMENSION' then the second element of the array is a string representing a dimension name. If the type enumeration is 'TYPE_NONE' then the second element will not be present, indicating that this cell has no PickList.

TM1ViewArrayValuePickListByRangeGet

Item	Description
Purpose	Call this function to check if a PickList exists or not for a given range of cells.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayValuePickListByRangeGet(TM1P hPool, TM1V hView, TM1V iRowStart, TM1V iColStart, TM1V iRowEnd, TM1V iColEnd)</pre>
Parameters	hView - view that contains the cell(s) you wish to check for existence of a PickList. iRowStart - starting cell's row in the view. iColStart - starting cell's column in the view. iRowEnd - ending cell's row in the view. iColEnd - ending cell's column in the view.
Result	A two element array. The first element is an array of pick list values. The second element is an array of indices into the array of pick list values for each cell in the range.

TM1ViewArrayValuePickListExists

Item	Description
Purpose	Call this function to check if a PickList exists or not for a given cell.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewArrayValuePickListExists(TM1P hPool, TM1V hView, TM1V iColumn, TM1V iRow)</pre>
Parameters	hView - view that contains the cell you wish to check for existence of a PickList iColumn - the cell's column in the view iRow - the cell's row in the view
Result	A Boolean indicating whether or not a PickList exists for this cell.

TM1ViewCellValueGet

Item	Description
Purpose	Retrieves a single value from a cell in a view.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewCellValueGet(TM1P hPool, TM1V hView, TM1V hArrayOfElements);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. <i>hView</i> is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views can be retrieved using the list properties of TM1CubeViews. hArrayOfElements is a TM1V containing an array of element handles.
Result	Returns a TM1V that contains the data in an IBM Cognos TM1 view cell.
Security	None.
Errors	None.
See Also	Other TM1View functions.

TM1ViewCellsValueGet

Item	Description
Purpose	Retrieves a values from a range of cells in a view.
Definition	<code>TM1V vValues = TM1ViewCellsValueGet(TM1PhPool, TM1V hView, TM1V vArrayOfCells)</code>
Parameters	<i>hPool</i> is a pool handle obtained with <code>TM1ValPoolCreate</code>. <i>hView</i> is a handle to a view. A handle to a new view is returned by the function <code>TM1ViewCreate</code>. Handles to existing views can be retrieved using the list properties of <code>TM1CubeViews</code>. <i>vArrayOfCells</i>: array of cells, each cell is an array of coordinate of element handles in the exact same order as the dimension in the cube.
Result	An array of cell raw values, each array contains an array of cell values, has spread hold status, etc.
Example	<pre>Array(0) // cell 1 Array(0)element 1 handle Array(1)element 2 handle ... Array(n-1)element n handle Array(1) // cell 2 ... Array(n-1) // n number of cells</pre>

TM1ViewCreate

Item	Description
Purpose	Creates a view for a cube. A view is implemented as a sub-object of a cube.
Definition	<code>TM1IMPORT TM1V TM1API TM1ViewCreate(TM1P hPool, TM1V hCube, TM1V hTitleSubsetArray, TM1V hColumnSubsetArray, TM1V hRowSubsetArray)</code>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hCube is a valid cube handle obtained with TM1CubeCreate or from one of the TM1ListObject functions. hTitleSubsetArray is a handle to an array of subset handles. In IBM Cognos TM1 Perspectives, this list of subsets appears under the static dimensions in a view. If there is more than one title dimension, the array will have more than one element.
	This is a view of the 94sales cube. To create this view, you would create two subsets: A subset of the actvsbud dimension, one element of which is Variance, as displayed in the example. A subset of the region dimension, one element of which is World, as displayed in the example. Create an array that contains handles to these two subsets, then pass a handle to that array as the hTitleSubsetArray argument. hColumnSubsetArray is a handle to an array of subset handles. These subsets are displayed along the columns of the view. In the example, two subsets are passed: one subset of the month dimension, and one subset of the model dimension. hRowSubsetArray is a handle to an array of subset handles. These subsets are displayed in the rows of the view. In the example, a subset of the account1 dimension is passed.
Result	Creates a new view, and returns a handle to it. If all the elements of a dimension are to be used for the view do not use a subset handle. Use the handle of the subset's dimension instead.
Security	None.
Errors	None.
See Also	Other TM1View functions

TM1ViewExtractCreate

Item	Description
Purpose	Creates a sequential list of records from a view.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewExtractCreate(TM1PhPool, TM1V hView);</pre>

Item	Description
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views can be retrieved using the list properties of TM1CubeViews.
Result	Return a TM1V containing a handle to the view extract. (view extract object type is TypeOldQuery).
Security	None.
Errors	None.
See Also	TM1ViewExtractDestroy TM1ViewExtractGetNext

TM1ViewExtractDestroy

Item	Description
Purpose	Destroys a view extract created by TM1ViewExtractCreate.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewExtractDestroy(TM1P hPool, TM1V hExtract);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hExtract is a handle to an extract. A handle to an extract is returned by the function TM1ViewExtractCreate.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the function executed successfully.
Security	None.
Errors	None.
See Also	TM1ViewExtractCreate TM1ViewExtractGetNext

TM1ViewExtractGetNext

Item	Description
Purpose	Return the result of a view extract.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1ViewExtractGetNext(TM1PhPool, TM1V hExtract);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hExtract is a handle to an extract. A handle to an extract is returned by the function TM1ViewExtractCreate.
Result	Returns a TM1V. Element positions are returned as indexes, element. Names are returned as strings, and values are returned as reals or strings.
Security	None.
Errors	None.
See Also	TM1ViewExtractCreate TM1ViewExtractDestroy

TM1ViewListByNamesGet

Item	Description
Purpose	Gets an array of view objects for the specified cube and view names.
Definition	<pre>TM1V hCube, TM1v vViewNames, TM1VbFlag</pre>
Parameters	hCube: a cube handle vViewNames: an Array of view names bFlag: whether to get only public views, (default would be private views first, if not exist, then public views)
Result	vViewList: an Array of views for the specified cube, each array includes view handle, name, and perhaps other properties to be determined. Array (): Array(VL_VIEWHANDLE): Array(VL_VIEWNAME): Array(VL_ISPRIVATE): ...

TM1ViewListGet

Item	Description
Purpose	Gets an array of view objects for the specified cube.
Definition	TM1V hCube, TM1V iFlag
Parameters	hCube: a cube handle iFlag: private views/public views/both (VL_GET_PRIVATEDATA VL_GET_PUBLICDATA)
Result	vViewList: an Array of views for the specified cube, each array includes view handle, name, and other properties to be determined. Array (): Array(VL_VIEWHANDLE): Array(VL_VIEWNAME): Array(VL_ISPRIVATE): ...

Data Spreading Syntax

You can use spreading control codes with a set syntax.

The following table provides information about the spreading control code argument in the TM1CubeCellSpreadViewArray and TM1CubeCellSpread functions. This code is a string built from the elements described in the following table.

Data Spreading Method	Code	Data Action (Optional) Δ	Direction Indicators	Required Method Parameters	Example
Proportional Spread	P	+, ~	, ^, <, >	Value to be spread	P<>100
The previous example proportionally spreads the value 100 to all leaf cells on the row of insertion, replacing existing cell values.					
Equal Spread	S	+, ~	, ^, <, >	Value to be spread	S+ ^200
The previous example equally spreads the value 200 to all leaf cells on the column of insertion, adding the product of spreading to existing cell values.					
Repeat	R	+, ~	, ^, <, >	Value to be spread	R~<50

Data Spreading Method	Code	Data Action (Optional) ∆	Direction Indicators	Required Method Parameters	Example
The previous example subtracts the value 50 from all leaf cells left of the insertion point.					
Percent Change	P%	+, ~	, ^, <, >	Percentage	P%+ ^<>10
The previous example applies a percent change of 10% to all leaf values in the view and adds the product to existing cell values.(It increments all leaves in the view by 10%.)					
Straight Line	SL	+, ~	, ^, <, > ∆∆	Start Value and End Value	SL>100:200
The previous example applies Straight Line spreading to replace all leaf values right of the point of insertion, using a start value of 100 and an end value of 200.					
Growth %	GR	+, ~	, ^, <, > ∆∆	Start Value and Growth Percentage	GR 300:25
The previous example applies a 25% growth percentage to the starting value of 300 and replaces all leaf values below the point of insertion.					
Clear	C	N/A	, ^, <, >	N/A	C ^<>
The previous example clears values from all cells in the view.					
Leaf Hold	H	N/A	, ^, <, >	N/A	H<>
The previous example holds all leaf cells on the row of insertion.					
Release Leaf Hold	RH	N/A	, ^, <, >	N/A	RH<>
The previous example releases all leaf holds on the row of insertion.					
Consolidation Hold	HC	N/A	, ^, <, >	N/A	HC<>
The previous example holds all consolidated cells on the row of insertion.					
Release Consolidation Hold	RC	N/A	, ^, <, >	N/A	RC<>
The previous example releases all consolidated cells on the row of insertion.					
* The default data action is Replace. The spreading syntax uses the tilde character (~) to denote the Subtract data action, and the plus symbol (+) to denote the Add data action.					

Data Spreading Method	Code	Data Action (Optional) △	Direction Indicators	Required Method Parameters	Example
** Straight Line and Growth % methods can be used across a single row or column. Rectangular ranges are not allowed. Direction combinations of up and down (^) or left and right(<>) are the only combinations allowed for these spreading methods.					

Chapter 5. IBM Cognos TM1 functions for Microsoft Visual Basic

This section contains a complete description of the functions contained in the IBM Cognos TM1 API for Microsoft Visual Basic.

The functions are presented in alphabetical order.

Function types and naming conventions

There are several groups of functions included in the IBM Cognos TM1 API.

The API uses an ordered naming system to separate the functions into groups.

All function names begin with TM1, followed by the object and sub-object to which the function applies, optionally followed at the end by a verb that describes the action taken. For example:

TM1ClientGroupAssign

The API functions can be grouped as follows:

- System functions, which are used to interact with the API itself and do not involve interaction with the Servers start with the prefix TM1System.
- Functions that are used to operate on value capsules start with the prefix TM1Val.
- Functions that operate on all objects start with the prefix TM1Object.
- Functions that are used to operate on servers, cubes, dimensions and other objects start with the prefix TM1ObjectType. For example:

Server objects start with the prefix TM1Server.

Cube objects start with the prefix TM1Cube.

Dimension objects start with the prefix TM1Dimension, and so on.

TM1APIInitialize

Initializes the IBM Cognos TM1 API.

Item	Description
Purpose	Initializes the IBM Cognos TM1 API. Call this function at the beginning of your TM1 application.
Definition	Declare Sub TM1APIInitialize Lib "tmlapi.dll" ()
Parameters	None.

Item	Description
Result	This function performs per-process initialization for the TM1 API. This function was added to avoid some memory conflicts that can occur in very complicated TM1 Applications that involve multiple users. You should call this function at the beginning of your TM1 API application. This function is part of the API initialization sequence required by every TM1 API program. For more information, see “Connecting to the API” on page 5. You should call TM1APIFinalize at the end of your TM1 API application.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorClientAlreadyExists TM1ErrorObjectSecurityNoAdminRights
See Also	TM1APIFinalize

TM1APIFinalize

Cleans up memory structures used during IBM Cognos TM1 API processing.

Item	Description
Purpose	Cleans up memory structures used during IBM Cognos TM1 API processing.
Definition	Declare Function TM1APIFinalize Lib "tm1api.dll" ()
Parameters	None.
Result	This function cleans up memory locks and performs other cleanup for the TM1 API. This function is part of the cleanup and logout sequence required for a well-behaved TM1 API program. For more information, see “Disconnecting from the API” on page 8. You should call TM1APIInitialize at the beginning of your TM1 API application.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	TM1APIInitialize

TM1BlobClose

Closes the BLOB

Item	Description
Purpose	Closes the BLOB. When finished with reading or writing, you should call this function to close the BLOB.
Definition	Declare Function TM1BlobClose Lib "tm1api.dll" (ByVal hPool As Long, ByVal hBlob As Long) As Long
Parameters	hPool is a TM1Val. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hBlob is a TM1Val. This TM1 value capsule contains the handle of the BLOB. You can retrieve this handle from the TM1 server list property TM1ServerBlobs.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None.
Errors	TM1ErrorBlobCloseFailed
See Also	Other TM1Blob functions.

TM1BlobCreate

Creates a BLOB with the specified name and registers the object on the server.

Item	Description
Purpose	Creates a BLOB with the specified name and registers the object on the server. You don't need to make a separate registration call to register the object.
Definition	Declare Function TM1BlobCreate Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal sName As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a TM1Val. This TM1 value capsule contains the handle of the TM1 server on which the BLOB will be created. sName is a TM1Val containing a TM1_STRING. This string is the name of the BLOB to be created.

Item	Description
Result	The function returns a TM1 value capsule containing the handle to the BLOB created.
Security	None.
Errors	TM1ErrorBlobCreateFailed
See Also	Other TM1Blob functions.

TM1BlobGet

Retrieves bytes of data from the BLOB.

Item	Description
Purpose	Retrieves n bytes of data from the BLOB starting at location x. The data is returned in the argument buf.
Definition	Declare Function TM1BlobGet Lib "tm1api.dll" (ByVal hUser As Long, ByVal hBlob As Long, ByVal x As Long, ByVal n As Long, ByVal buf As String) As Long
Parameters	hUser is the user handle obtained with TM1SystemOpen. hBlob is a TM1Val. This IBM Cognos TM1 value capsule contains the handle of the BLOB. You can retrieve this handle from the TM1 server list property TM1ServerBlobs. x is a TM1_INDEX. This integer is the starting location in the BLOB to retrieve data from. n is a TM1_INDEX. This integer is the number of bytes to retrieve. buf is a string. The data retrieved from the BLOB is written to this string when the TM1BlobGet function successfully completes.
Result	Returns a TM1 value capsule containing the number of bytes successfully read from the BLOB.
Security	None.
Errors	None.
See Also	Other TM1Blob functions.

TM1BlobOpen

Opens the BLOB for READ or WRITE.

Item	Description
Purpose	Opens the BLOB for READ or WRITE.
Definition	Declare Function TM1BlobOpen Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hBlob As Long) As Long
Parameters	hPool is a TM1Val. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hBlob is a TM1Val. This TM1 value capsule contains the handle of the BLOB. You can retrieve this handle from the TM1 server list property TM1ServerBlobs.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean.
Security	None.
Errors	TM1ErrorBlobOpenFailed
See Also	Other TM1Blob functions.

TM1BlobPut

Writes data to a BLOB.

Item	Description
Purpose	Writes data to a BLOB.
Definition	Declare Function TM1BlobPut Lib "tmlapi.dll" (ByVal hUser As Long, ByVal hBlob As Long, ByVal x As Long, ByVal n As Long, ByVal buf As String) As Long
Parameters	hUser is the user handle obtained with TM1SystemOpen. hBlob is a TM1Val. This IBM Cognos TM1 value capsule contains the handle of the BLOB. You can retrieve this handle from the TM1 server list property TM1ServerBlobs. x is a long. This index is the starting location to which you want to write the contents of <i>buf</i>. n is a long. This index is the number of bytes to be written to the BLOB. buf is a string. This string contains the data to be written.

Item	Description
Result	The function returns the number of bytes written successfully.
Security	None.
Errors	None.
See Also	Other TM1Blob functions.

TM1ChoreExecute

Executes a TurboIntegrator chore.

Item	Description
Purpose	Executes a TurboIntegrator chore on an IBM Cognos TM1 server.
Definition	<pre>Declare Function TM1ChoreExecute Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hChore As Long) As Long</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hChore is a value capsule containing a valid handle to a chore defined on the TM1 Server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerChores.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 0, the chore execution generated errors, otherwise the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
Security	None
Errors	None
See Also	TM1ProcessExecute

TM1ClientAdd

Adds a new client to a server.

Item	Description
Purpose	Adds a new client to a server.

Item	Description
Definition	<pre>Declare Function TM1ClientAdd Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal sClientName As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to the server to which the client will be added. sClientName is a long. This TM1 value capsule contains a string which is the name of the client to be added to the server.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. After calling TM1ClientAdd you must assign a password to the client with the function TM1ClientPasswordAssign. After adding a client, call TM1ObjectListHandleByNameGet to get a handle to the client. It is strongly suggested that you assign a password to the client with the function TM1ClientPasswordAssign after adding a new client.
Security	The client must have ADMIN rights to the server.
Errors	<pre>TM1ErrorClientAlreadyExists TM1ErrorObjectSecurityNoAdminRights</pre>
See Also	Other TM1Client functions.

TM1ClientGroupAssign

Assigns a client to a group.

Item	Description
Purpose	Assigns a client to a group.
Definition	<pre>Declare Function TM1ClientGroupAssign Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hClient As Long, ByVal hGroup As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hClient is a long. This TM1 value capsule contains a handle to the client to be assigned to the group. To retrieve the client handle from the server call TM1ObjectListHandleByNameGet. The format of this function is as follows: <pre>vResult = TM1ObjectListHandleByNameGet(hPool, TM1ServerClients(), vClientName)</pre> hGroup is a long. This TM1 value capsule contains a handle to the group to which the client is to be assigned. To retrieve the group handle from the server, call TM1ObjectListHandleByNameGet. The format of this function is as follows: <pre>vResult = TM1ObjectListHandleByNameGet (hPool, TM1ServerGroups(), vGroupName)</pre>
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is 1, the operation was successful.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Client functions.

TM1ClientGroupsAssigned

Determines whether a client is assigned to a group.

Item	Description
Purpose	Determines whether a client is assigned to a group.
Definition	<pre>Declare Function TM1ClientGroupIsAssigned Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hClient As Long, ByVal hGroup As Long) As Long</pre>
Parameters	hPool is a long. This TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hClient is a long. This IBM Cognos TM1 value capsule contains a handle to a client. hGroup is a long. This TM1 value capsule contains a handle to a group.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Client functions.

TM1ClientGroupRemove

Removes a client from a group.

Item	Description
Purpose	Removes a client from a group.
Definition	Declare Function TM1ClientGroupRemove Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hClient As Long, ByVal hGroup As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hClient is a long. This TM1 value capsule contains a handle to a client. hGroup is a long. This TM1 value capsule contains a handle to a group.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful.
Security	The client must have ADMIN rights to the Server.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Client functions.

TM1ClientHasHolds

Checks whether the client has hold cells or not.

Item	Description
Purpose	Checks whether the client has hold cells or not.

Item	Description
Definition	<pre>TM1IMPORT TM1V TM1API TM1ClientHasHolds(TM1P hPool, TM1V hClient);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hClient is a client handle. Client handles are returned by the function TM1SystemServerConnect. You can also retrieve a client handle from the server list property TM1ServerClients.
Result	The function returns a TM1V containing a TM1_BOOL. If the boolean is TRUE, the client has one or more hold cells.
Security	None.
Errors	None.
See Also	Other TM1ViewArray functions.

TM1ClientPasswordAssign

Assigns a new password to a client.

Item	Description
Purpose	Assigns a new password to a client.
Definition	<pre>Declare Function TM1ClientPasswordAssign Lib "tm1api.dll" (ByVal hPool As Long, ByVal hClient As Long, ByVal sPassword As Long) As Long</pre>
Parameters	hPool is a long. This TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hClient is a long. This IBM Cognos TM1 value capsule contains a handle to a client object. To retrieve the client handle from the server call TM1ObjectListHandleByNameGet. The format of this function is as follows: <pre>vResult = TM1ObjectListHandleByNameGet(hPool, TM1ServerClients(), vClientName);</pre> The vClientName argument should be the name you added with TM1ClientAdd. sPassword is a long. This TM1 value capsule contains the password you are assigning to the client. You can construct this value capsule with the function TM1ValString.
Result	Returns a long. The long contains a value capsule. Pass the value capsule to TM1ValStringGet_VB to retrieve the new password.

Item	Description
Security	Only the client whose password is being changed and clients with ADMIN privileges can assign passwords.
Errors	None.
See Also	TM1ClientAdd

TM1ConnectionCheck

Checks a connection object for consistency.

Item	Description
Purpose	Checks a connection object for consistency.
Definition	Declare Function TM1ConnectionCheck Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hConnection As Long) As Long
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate hConnection is a handle to a connection object. This object handle is returned by TM1ConnectionCreate, or it can be retrieved from the IBM Cognos TM1 server List object TM1ServerConnections.
Result	The function returns a Boolean 1 if the operation is successful.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1ConnectionCreate

Creates a new connection object.

Item	Description
Purpose	Creates a new connection object.
Definition	Declare Function TM1ConnectionCreate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal sMasterServerName As Long, ByVal sUsername As Long, ByVal sPassword As Long) As Long

Item	Description
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle of the server on which the connection will be created. sMasterServerName is a string value containing the name of the star server. sUsername is a string value containing the name of the IBM Cognos TM1 client. sPassword is a string value containing the password.
Result	This function returns a handle to a connection object. The TM1 servers on either side of the connection must be registered with the TM1 ADMIN host that you specified when you called TM1SystemAdminHostSet. Once you have created the connection, you should populate the following connection object properties: TM1ConnectionSyncStarToPlanet - Data changed on the star server is migrated to the planet server during a synchronization. TM1ConnectionSyncPlanetToStar - Data changed on the planet server is migrated to the Star server during a synchronization.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1ConnectionDelete

Deletes a connection object.

Item	Description
Purpose	Deletes a connection object.
Definition	<pre>Declare Function TM1ConnectionDelete Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hConnection As Long) As Long</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hConnection is a handle to a connection object. This object handle is returned by TM1ConnectionCreate, or it can be retrieved from the IBM Cognos TM1 server List object TM1ServerConnections.
Result	This function deletes a connection object from the TM1 server.
Security	The client must have ADMIN rights to the server.

Item	Description
Errors	None.
See Also	Other TM1Connection functions.

TM1ConnectionSynchronize

Performs synchronization on a connection object.

Item	Description
Purpose	Performs synchronization on a connection object.
Definition	Declare Function TM1ConnectionSynchronize Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hConnection As Long) As Long
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate hConnection is a handle to a connection object. This object handle is returned by TM1ConnectionCreate, or it can be retrieved from the IBM Cognos TM1 server list property TM1ServerConnections.
Result	The function returns a Boolean 1 if the operation is successful. A successful synchronization means that cubes on both sides of the connection have the latest data.
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1Connection functions.

TM1CubeCellDrillListGet

Returns a list of drill object process names associated with a cell.

Item	Description
Purpose	Returns a list of drill object process names associated with a cell.
Definition	Declare Function TM1CubeCellDrillListGet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hArrayOfKeys As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a handle to a cube. This handle is returned either by TM1CubeCreate or by calling one of the TM1ObjectListHandle functions. hArrayOfKeys is a long. This TM1 value capsule contains an array of element handles. There should be one element handle for each dimension in the cube. These element handles, in combination, identify the exact cell whose list of drill processes you want to retrieve.
Result	The function returns a TM1V array which includes all the drill object process names.
Security	None.
Errors	TM1ErrorCubeDrillNotFound TM1ErrorCubeNumberOfKeysInvalid TM1ErrorCubeDrillInvalidStructure TM1ErrorSystemParameterTypeInvalid TM1ErrorCubeKeyInvalid
See Also	Other TM1ViewArray functions.

TM1CubeCellDrillObjectBuild

Returns a drill object and a drill object process name.

Item	Description
Purpose	Returns a drill object associated with a cell and a drill object process name.
Definition	Declare Function TM1CubeCellDrillObjectBuild Lib "tm1api.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hArrayOfKeys As Long, ByVal sDrillProcessName As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a handle to a cube. This handle is returned either by TM1CubeCreate or by calling one of the TM1ObjectListHandle functions. hArrayOfKeys is a long. This TM1 value capsule contains an array of element handles. There should be one element handle for each dimension in the cube. These element handles, in combination, identify the exact cell whose list of drill processes you want to retrieve. sDrillProcessName is a long. This TM1 value capsule contains a string. This is the string name of the drill object process.
Result	The TM1 server runs the drill object process and returns a TM1V containing an object. The object is one of the following types: TM1TypeSQLTable TM1TypeView If the returned object type is TM1TypeSQLTable, you can retrieve following properties: TM1SQLTableColumnNames TM1SQLTableColumnTypes TM1SQLTableNumberOfColumns TM1SQLTableNumberOfRows TM1SQLTableRowsetSize After using this object, you should delete it with TM1ObjectDestroy.
Security	None.
Errors	TM1ErrorObjectNotFound TM1ErrorSystemParameterTypeInvalid
See Also	TM1CubeCellDrillListGet

TM1CubeCellSpread

Spreads data to an array of cells in one or more cubes.

Item	Description
Purpose	Spreads data to an array of cells in one or more cubes. This function uses cube handles and element handles to mark the starting location for the spread command. No view handle is required for this function.

Item	Description
Definition	Declare Function TM1CubeCellSpread Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal vArrayOfCells As Long, ByVal vCellReference As Long, ByVal sSpreadData As Long) As Long
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect. vArrayOfCells is a TM1V containing an array cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1, Array2, Array3..., Arrayn} Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle} Array2 = {CubeHandle2, ElementHandle, ElementHandle..., ElementHandle}</pre> The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property TM1DimensionElements.
Parameters (cont.)	vCellReference is the reference cell for vArrayOfCells. Both vCellReference and vArrayOfCells must be single cell ranges. vCellReference may refer to a cell in any cube, but the target cell must be a consolidated cell, and the consolidation must be identical to the one referenced by vArrayOfCells. vCellReference is only used for Relative Proportional Spread and Relative Percent Adjustment. It is ignored in any other case. vCellReference is a TM1V containing a TM1 array. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. It has the general form [cubehandle, elementhandle1, elementhandle2, elementhandle3...]. To indicate that there is no reference cell, this parameter should be set to TM1ArrayNull or to an array with size zero. sSpreadData is a TM1V containing a string value. This string is the spreading command. For example, S>100. For a complete list of the TM1 spreading commands, please refer to "Spreading control codes" on page 410.

Item	Description
Result	Use this function to spread a value when the client application does not have a view handle available. For example, if you are spreading values in a spreadsheet that contains DBRW functions, you should use this function. This function ignores the direction codes in the control string. It is incumbent on the programmer to build the vArrayOfCells array with the correct cell range.
Result (cont.)	The function returns a TM1V containing three successful cases: TM1CubeCellSpreadFunctionOk() indicates the spread was performed successfully. TM1CubeCellSpreadNumericCellSetOk() indicates that the control string was a number and that it was successfully entered into the numeric cell. TM1CubeCellSpreadStringSetOk() indicates that the string was successfully entered into the string cell.
Security	None.
Errors	TM1ErrorDataSpreadFailed() TM1ErrorCubeCellWriteStatusCubeNoWriteAccess() TM1ErrorCubeCellWriteStatusCubeReserved() TM1ErrorCubeCellWriteStatusCubeLocked() TM1ErrorCubeCellWriteStatusRuleApplies() TM1ErrorCubeCellWriteStatusElementIsConsolidated() TM1ErrorCubeCellWriteStatusElementNoWriteAccess() TM1ErrorCubeCellWriteStatusElementReserved() TM1ErrorCubeCellWriteStatusElementLocked()
See Also	TM1CubeCellSpreadViewArray

TM1CubeCellSpreadStatusGet

Checks the status of the cells.

Item	Description
Purpose	Checks the status of the cells of an IBM Cognos TM1 view or a TM1 cube.

Item	Description
Definition	Declare Function TM1CubeCellSpreadStatusGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal hCells As Long, ByVal hCellRange As Long) As Long
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hServer is a handle to a TM1 server object. This handle is returned by the function TM1SystemServerConnect. hCells is a TM1V containing one of two values: A two dimensional array of cell references of the form: [[cubehandle1, elemhandle, elemhandle,...], [cubehandle2, elemhandle, elemhandle,...]] The cubehandles can refer to different cubes. In this case, the status of cells within multiple cubes is returned, and the hCellRange parameter is ignored. A handle to a view, which will be used to extract the range of cells, defined by hCellRange. The function returns the status of all the cells in this range. hCellRange is a TM1V containing one of the following values: NULL Object - If hCells is an array, hCellRange should be set to a NULL object.
Parameters	A TM1V containing an array. This argument is used only when hCells is a handle to a view. This is an array of indices indicating the upper left and lower right cells coordinates of a range within the view. It has the general form [column1, row1, column2, row2]. If column2, row2 are not specified then the function returns the status of the cell defined by [column1, row1] inside the view. TM1ArrayNull() or an array with zero elements then the function returns the status of all the cells in the view.
Result	The function returns a TM1V containing an array of indices. There is one element in the array for each cell specified in the hCells and hCellRange arguments. If hCells is an array of cells then the items in the returned array will match the ones in hCells. If hCells is a view handle then the items in the array correspond to the cells in the view range as shown in the figure following the table.

Item	Description
Result (cont.)	Each value in the returned array is one of the following: TM1CubeCellSpreadStatusHeld indicates the cell is being held and will be ignored when included in all the spreading operations except RELEASE and RELEASE ALL. TM1CubeCellSpreadStatusHeldConsolidation indicates the cell's value will not be affected when this cell is included in a spreading function. (Consolidated values are not directly changed by spreading data. They may be recalculated if their component leaf cells are modified by the spreading function.) TM1CubeCellSpreadStatusWritable indicates the cell's value will be affected when this cell is included in a spreading function.
Security	None.
Errors	TM1ErrorSystemValueInvalid TM1ErrorObjectNotLoaded TM1ErrorViewNotConstructed TM1ErrorSystemValueInvalid TM1ErrorSystemParameterTypeInvalid
See Also	TM1CubeCellSpreadViewArray TM1CubeCellSpread

The following figure shows the target cell range for the spreading operation in blue. The numbers in parenthesis are the indices where the spreading status for those cells will be in the returned array.

		Region					
Account	Month	Denmark	Norway	Sweden	Belgium	Netherlands	Germany
Sales	1Quarter	4000	600	900	600	7000	6000
	Jan	900	600 (0)	700 (1)	300 (2)	888 (3)	890
	Feb	7900	8890 (4)	9000 (5)	7700 (6)	700 (7)	789
	March	677	6886 (8)	3243 (9)	67 (10)	7676 (11)	8000

TM1CubeCellSpreadViewArray

Spreads data specified in sControl.

Item	Description
Purpose	Spreads data specified in sControl to a range of cells in a view. This function uses row and column pairs to mark the starting location for the spread command.
Definition	<pre>Declare Function TM1CubeCellSpreadViewArray Lib "tm1api.dll" (ByVal hPool As Long, ByVal hView As Long, ByVal aCellRange As Long, ByVal aCellRef As Long, ByVal sControl As Long) As Long</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews. aCellRange is a handle to an array value. This array contains the locations in the view to which your data will be spread. This array can contain either two values or four values. If the array contains two TM1V integer values (column1, row1), the specified cell is used as a starting point for the data spread. The remainder of the range is determined by the sControl argument. If the array contains four TM1V integer values (column1, row1, column2, row2), the paired coordinates represent the starting and ending cells of the range where the data will be spread. aCellRef is only used for Relative Proportional Spread and Relative Percent Adjustment. This TM1V contains an IBM Cognos TM1 array. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. To indicate that there is no reference cell, this parameter should be set to TM1ArrayNull() or to an array with size zero.
Parameters (cont.)	aCellRef is the reference cell for aCellRange. Both aCellRef and aCellRange must be single cell ranges. aCellRef may refer to a cell in any cube, but the target cell must be a consolidated cell, and the consolidation must be identical to the one referenced by aCellRange. sControl is a TM1V containing a string value. This string is the spreading command. For example, S>100. For a complete list of the TM1 spreading commands, please refer to "Spreading control codes" on page 410.
Result	The function returns a TM1V containing three successful cases: TM1CubeCellSpreadFunctionOk() TM1CubeCellSpreadNumericCellSetOk() TM1CubeCellSpreadStringCellSetOk()

Item	Description
Security	None.
Errors	TM1ErrorDataSpreadFailed TM1ErrorObjectNotLoaded TM1ErrorViewNotConstructed
See Also	TM1CubeCellSpread

TM1CubeCellValueGet

Retrieves the value of a cell from a cube.

Item	Description
Purpose	Retrieves the value of a cell from a cube.
Definition	Declare Function TM1CubeCellValueGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hArrayOfElements As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a handle to the cube from which the data will be retrieved. hArrayOfElements is a long. This TM1 value capsule contains an array of element handles, one from each of the dimensions of the cube, in the same order as the dimensions themselves.
Result	Returns the value stored in the cell specified.
Security	The client must have at least READ access to the cube, and to all the elements that identify the cell.
Errors	TM1ErrorCubeKeyInvalid TM1ErrorCubeNumberOfKeysInvalid TM1ErrorObjectSecurityNoReadRights
See Also	TM1CubeCellValueSet

TM1CubeCellValueSet

Updates the value of a cell in a cube.

Item	Description
Purpose	Updates the value of a cell in a cube.

Item	Description
Definition	<pre> Declare Function TM1CubeCellValueSet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hArrayOfElements As Long, ByVal hValue As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a handle to the cube from which the data will be retrieved. hArrayOfElements is a long. This TM1 value capsule contains an array of element handles, one from each of the dimensions of the cube, in the same dimension order as that with which the cube is defined. hValue is a long. This TM1 value capsule contains the value to be stored in the cell.
Result	Returns the new value of the cell. Compare the returned TM1 value capsule to one of the following constants: Tm1ValTypeError - If the IBM Cognos TM1 value capsule contains an error, the cell was not updated successfully. Tm1ValTypeString - If the TM1 value capsule contains a string, the cell now contains a new string value. Tm1ValTypeReal - If the TM1 value capsule contains a real number, the cell now contains a new numerical value. Tm1ValTypeBool - If the TM1 value capsule contains a TM1 Boolean type, and the Boolean equals 0, the cell was not updated successfully.
Security	The client must have at least WRITE access to the cube and to all the dimensions that identify the cell.
Errors	TM1ErrorCubeKeyInvalid TM1ErrorCubeNumberOfKeysInvalid TM1ErrorCubeCellValueTypeMismatch TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectSecurityNoWriteRights
See Also	TM1CubeCellValueGet

TM1CubeCreate

Creates a new cube.

Item	Description
Purpose	Creates a new cube.

Item	Description
Definition	Declare Function TM1CubeCreate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal hArrayOfDimensions As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to the server on which to create the cube. hArrayOfDimensions is a long. This TM1 value capsule contains a handle to an array of between 2 and 16 dimension handles with which to construct the cube. Specify free dimensions by setting the corresponding element handle to zero.
Result	The function returns a handle to the newly created cube. The cube must be registered before other applications can access it.
Security	The client must have at least READ access to the dimensions used to create the cube.
Errors	TM1ErrorCubeDimensionInvalid TM1ErrorCubeNotEnoughDimensions TM1ErrorCubeTooManyDimensions TM1ErrorObjectSecurityNoReadRights TM1ErrorCubeCreationFailed
See Also	TM1DimensionCreateEmpty

TM1CubePerspectiveCreate

Calculates a perspective of a cube.

Item	Description
Purpose	Calculates a perspective of a cube. A perspective is a sub-cube of a cube. It is defined by choosing one or more free dimensions, which will be the dimensions of the resulting sub-cube. The rest of the dimensions are fixed by choosing a specific element from each.
Definition	Declare Function TM1CubePerspectiveCreate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hArrayOfElementTitles As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a handle to the cube from which the perspective will be built. hArrayOfElementTitles is a long. This TM1 value capsule contains an array of element handles, one from each of the dimensions of the cube, in the dimension order of the cube. Specify free dimensions by setting the corresponding element handle to zero.
Result	The function returns a handle to the perspective generated. The perspective created is stored with the cube. Any reference to a cell in the perspective will be satisfied from the perspective.
Security	The client must have at least READ access to the cube, and to all the fixed elements.

TM1CubePerspectiveDestroy

Deletes a perspective of a cube.

Item	Description
Purpose	Deletes a perspective of a cube. A perspective is a sub-cube of a cube. It is defined by choosing one or more free dimensions, which will be the dimensions of the resulting sub-cube. The rest of the dimensions are fixed by choosing a specific element from each.
Definition	<pre> Declare Function TM1CubePerspectiveDestroy Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hArrayOfElementTitles As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a handle to the cube from which the perspective will be deleted. hArrayOfElements is a long. This TM1 value capsule contains an array of element handles, one from each of the dimensions of the cube, in the dimension order of the cube. Specify free dimensions by setting the corresponding element handle to zero.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function deletes the perspective.
Security	The client must have at least READ access to the cube.
Errors	TM1ErrorCubeNumberOfKeysInvalid
See Also	TM1CubePerspectiveCreate

TM1CubeShowsNulls

Returns whether the cube has the UNDEFVALS rule.

Item	Description
Purpose	Returns whether the cube has the UNDEFVALS rule.
Definition	TM1CubeShowsNulls(TM1P hPool, TM1V hCube);
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hCube is a Cube.
Result	The function returns a TM1V containing a TM1_BOOL. If the Boolean is 1, the cube has the UNDEFVALS rule defined. The default behavior of IBM Cognos TM1 cubes is to treat zeros as equivalent to nulls: zeros are not stored in the cube, and empty locations are displayed as zero. The UNDEFVALS rule, if present on the cube, causes the the cube to distinguish zeros and nulls, treating zeros as regular numeric data. An UNDEFVALS cube will store zero values, and will display blanks for empty locations.
Errors	None Except for the ParameterTypeInvalid error that results if any of the object APIs are called with the wrong object type
See Also	TM1CubeCellValueUndefined

TM1DimensionCheck

Checks a dimension for consistency.

Item	Description
Purpose	Checks a dimension for consistency.
Definition	Declare Function TM1DimensionCheck Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hDimension As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hDimension is a long. This TM1 value capsule contains a handle to the dimension to be checked.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the dimension has consistency and can be registered on the server. This function cannot be used with registered dimensions.
Errors	TM1ErrorDimensionHasCircularReferences TM1ErrorDimensionHasNoElements TM1ObjectIsRegistered
See Also	All TM1Dimension functions.

TM1DimensionCreateEmpty

Creates an empty dimension.

Item	Description
Purpose	Creates an empty dimension.
Definition	Declare Function TM1DimensionCreateEmpty Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to the server in which to create the dimension.

Item	Description
Result	The function returns a handle to the empty dimension object. When you create a new dimension, this is the first function you call. The complete sequence for creating a registered dimension is as follows: Call TM1DimensionCreateEmpty. This function returns a handle to an empty dimension. Populate the dimension with simple elements by calling TM1DimensionElementInsert. Add consolidated elements by calling TM1DimensionElementComponentAdd. Verify the integrity of the new dimension after the dimension has been populated by calling TM1DimensionCheck. Register the dimension with TM1ObjectRegister if the integrity is intact.
See Also	All TM1Dimension functions.

TM1DimensionElementComponentAdd

Adds a component to a consolidated element.

Item	Description
Purpose	Adds a component to a consolidated element.
Definition	<pre> Declare Function TM1DimensionElementComponentAdd Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hElement As Long, ByVal hComponent As Long, ByVal rWeight As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hElement is a long. This TM1 value capsule contains a handle to the consolidated element to which the component will be added. hComponent is long. This TM1 value capsule contains a handle to the element to be added as a component. rWeight is a real long. This TM1 value capsule contains a handle to the weight of the component. The default is 1.
Result	Returns a long integer. The long contains a value capsule. Pass value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The new component is inserted in the dimension. This function cannot be used with registered dimensions.

Item	Description
Security	Because the dimension being changed is unregistered, no security considerations apply.
Errors	TM1ErrorDimensionElement ComponentAlreadyExists TM1ErrorDimensionElement ComponentNotNumeric TM1ErrorDimensionCircularReferences TM1ErrorDimensionElementNotConsolidated TM1ErrorObjectIsRegistered
See Also	All TM1Dimension functions.

TM1DimensionElementComponentDelete

Deletes a component of a consolidated element.

Item	Description
Purpose	Deletes a component of a consolidated element.
Definition	Declare Function TM1DimensionElementComponentDelete Lib "tm1api.dll" (ByVal hPool As Long, ByVal hCElement As Long, ByVal hElement As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCElement is a long. This TM1 value capsule contains a handle to the consolidated element from which the component will be deleted. hElement is a long. This TM1 value capsule contains a handle to the element to delete from the consolidated element.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The component is deleted from the dimension. This function cannot be used with registered dimensions.
Security	The client must have WRITE access to the dimension.
Errors	TM1ErrorDimensionElement ComponentDoesNotExist TM1ErrorDimensionElementNotConsolidated TM1ErrorObjectIsRegistered TM1ObjectSecurityNoWriteRights
See Also	All TM1Dimension functions.

TM1DimensionElementComponentWeightGet

Retrieves the weight of a component of a consolidated element.

Item	Description
Purpose	Retrieves the weight of a component of a consolidated element.
Definition	Declare Function TM1DimensionElementComponentWeightGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hCElement As Long, ByVal hElement As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCElement is a long. This TM1 value capsule contains a handle to a consolidated element. hElement is a long. This TM1 value capsule contains a handle to the component within the consolidated element whose weight is sought.
Result	The function returns a real value. This value is the weight of the component in the consolidation. The default weight of a component is 1.
Errors	TM1ErrorDimensionElement ComponentDoesNotExist TM1ErrorDimensionElement1NotConsolidated
See Also	Other TM1Dimension functions.

TM1DimensionElementDelete

Deletes an element from a dimension.

Item	Description
Purpose	Deletes an element from a dimension.
Definition	Declare Function TM1DimensionElementDelete Lib "tm1api.dll" (ByVal hPool As Long, ByVal hElement As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hElement a long. This TM1 value capsule contains a handle to an element in the dimension.

Item	Description
Result	The function deletes all instances of the element from the dimension. For example, if the element appears in two different consolidations in the same dimension, both instances are deleted. This function can only be performed on unregistered dimensions. To delete an element from an existing dimension, follow these steps: Get the handle to the dimension you want to update. Typically, you would use a TM1ObjectListHandle call to do this. Make a copy of the dimension with TM1ObjectDuplicate. Delete the unwanted element from the copy with TM1DimensionElementDelete. Call TM1DimensionUpdate to replace the old dimension with the new one.
Security	The client must have WRITE access to the dimension.
Errors	TM1ErrorObjectIsRegistered TM1ErrorDimensionElementDoesNotExist TM1ErrorObjectSecurityNoWriteRights
See Also	TM1DimensionElementInsert

TM1DimensionElementInsert

Inserts an element in a dimension.

Item	Description
Purpose	Inserts an element in a dimension.
Definition	<pre> Declare Function TM1DimensionElementInsert Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hDimension As Long, ByVal hElementAfter As Long, ByVal sName As Long, ByVal iType As Long) As Long </pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hDimension is a long. This TM1 value capsule contains a handle to the dimension into which the element is inserted. hElementAfter is a long. This TM1 value capsule contains a handle to the element after which the new element is inserted. If the handle is TM1ObjectNull, the new element is inserted at the beginning of the dimension. sName is a long. This TM1 value capsule contains a handle to a string that specifies the name of the element. iType is a long. This TM1 value capsule contains a handle to an integer specifying the type of element. It can be: TM1TypeElementSimple() TM1TypeElementConsolidated() TM1TypeElementString() TM1TypeElement() will bring up one of the three elements listed above.
Result	The function returns a handle to the inserted element if the operation is successful. The new component is inserted in the dimension. This function cannot be used with registered dimensions. To update registered dimensions, follow these steps:
	Create a null handle. temp = TM1ObjectNull() Make an unregistered copy. hDupDim = TM1ObjectDuplicate (hPool, hDimension) Call the function.
	hElement = TM1DimensionElementInsert (hPool, hDupDim, temp, TM1ValString (hPool, "string", 0), TM1TypeElementSimple()) where Simple is Consolidated, String or Simple Call TM1DimensionUpdate to overwrite the registered dimension with the newly unregistered dimension.
	TM1DimensionUpdate (hPool, hDimension, hDupDim) Depending on the order in which you define the elements, they expand slightly differently.

Item	Description
Security	Since the dimension being changed is unregistered, no security considerations apply.
Errors	TM1ErrorDimensionElementAlreadyExists TM1ErrorObjectIsRegistered
See Also	All TM1Dimension functions.

TM1DimensionUpdate

Replaces a registered dimension with a new one.

Item	Description
Purpose	Replaces a registered dimension with a new one and updates all associated cubes.
Definition	Declare Function TM1DimensionUpdate Lib "tm1api.dll" (ByVal hPool As Long, ByVal hOldDimension As Long, ByVal hNewDimension As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hOldDimension is a long. This TM1 value capsule contains a handle to the registered dimension to be replaced. hNewDimension is a long. This TM1 value capsule contains a handle to the dimension that replaces the old one. The new dimension must be checked with the function TM1DimensionCheck before you call TM1DimensionUpdate.
Result	The function returns a handle to the updated dimension if the operation is successful. The old dimension is destroyed and replaced with the new one. All affected cubes are updated accordingly.
Security	The client must have ADMIN rights to the dimension being updated.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorDimensionNotChecked TM1ErrorObjectSecurityNoAdminRights
See Also	TM1DimensionCheck

TM1GroupAdd

Adds a new group to a server.

Item	Description
Purpose	Adds a new group to a server.
Definition	Declare Function TM1GroupAdd Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal sGroupName As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is long. This TM1 value capsule contains a handle to the server to which the group will be added. sGroupName is long. This TM1 value capsule contains a handle to a string containing the name of the group to be added.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function creates a new group on the server. To add clients to the new group, call TM1ClientGroupAssign.
Security	The client must have ADMIN rights to the server.
Errors	TM1ErrorGroupAlreadyExists TM1ErrorGroupMaximumNumberExceeded TM1ErrorObjectSecurityNoAdminRights
See Also	TM1ClientGroupAssign Other TM1Client functions.

TM1ObjectAttributeDelete

Deletes an attribute from an object and its siblings.

Item	Description
Purpose	Deletes an attribute from an object and its siblings.
Definition	Declare Function TM1ObjectAttributeDelete Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal hAttribute As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object from which the attribute is be deleted. hAttribute is a long. This TM1 value capsule contains a handle to the attribute to be deleted.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The attribute is deleted from the object and its siblings.
Security	The client must have ADMIN rights to the parent of the object.
Errors	TM1ErrorObjectAttributeDoesNotExist TM1ErrorObjectIsSecurityNoAdminRights TM1ErrorObjectIsUnregistered
See Also	All TM1Object functions.

TM1ObjectAttributeInsert

Inserts an attribute in an object and its siblings

Item	Description
Purpose	Inserts an attribute in an object and its siblings. Also used to create an alias attribute for an object.
Definition	<pre>Declare Function TM1ObjectAttributeInsert Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal hAttributeBefore As Long, ByVal sName As Long, ByVal sType As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the registered object for which the attribute is to be created. Use TM1ServerProperty to get a registered object. For example, use TM1ServerDimensions to get a Dimension, TM1ServerCubes to get a Cube. hAttributeBefore is long. This TM1 value capsule contains a handle to the attribute before which the new attribute is to be inserted. If the handle is TM1ObjectNull, the new attribute is inserted after the last attribute in the list. sName is a long. This TM1 value capsule contains a string that specifies the name of the attribute. sType is a long integer. This TM1 value capsule contains a handle to the string that specifies the type of attribute. It can be one of the following: TM1TypeAttributeNumeric() TM1TypeAttributeString() TM1TypeAttributeAlias()
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The new attribute is created for the object and its siblings.
Security	The client must have ADMIN rights to the parent of the object.
Errors	TM1ErrorObjectAttributeAlreadyExists TM1ErrorObjectIsUnregistered TM1ErrorObjectIsSecurityNoAdminRights
See Also	All TM1Object functions.

TM1ObjectAttributeValueGet

Retrieves the value of an attribute for any object.

Item	Description
Purpose	Retrieves the value of an attribute for any object. Also used to retrieve the value of an alias for the object specified.
Definition	Declare Function TM1ObjectAttributeValueGetLib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal hAttribute As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to an object. hAttribute is a long. This TM1 value capsule contains a handle to an attribute of the object. The possible attribute types are numeric, text, and alias.
Result	Returns the value of the attribute for the object. The type of the value depends on type of the attribute.
Security	The client must have READ access to the object in question in order to receive a result.
Errors	TM1ErrorObjectAttributeNotDefined TM1ErrorObjectSecurityNoReadRights TM1ObjectAttributeInsert
See Also	TM1ObjectAttributeValueSet

TM1ObjectAttributeValueSet

Updates the value of an object attribute.

Item	Description
Purpose	Updates the value of an object attribute. Also used to assign a name to the alias.
Definition	Declare Function TM1ObjectAttributeValueSetLib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal hAttribute As Long, ByVal hValue As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is long. This TM1 value handle contains a valid object. hAttribute is a long. This TM1 value capsule contains a handle to an attribute of the object. The possible attribute types are numeric, text, and alias. vValue is a long. This TM1 value capsule contains the value to be assigned to the attribute. The type of value depends on the type of the attribute that you are setting.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. An alias name must not be assigned for more than one object. More than one alias name, however, may be assigned to the same object.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectAttributeValueNotDefined TM1ErrorObjectAttributeTypeConflict TM1ErrorObjectAttributeAliasConflict
See Also	TM1ObjectAttributeValueGet TM1ObjectAttributeInsert

TM1ObjectAttributeValuesSet

Sets all element attributes at once.

Item	Description
Purpose	Sets all element attributes at once: TM1V TM1API TM1ObjectAttributeValuesSet(TM1P hPool, TM1V hAttribute, TM1V hObjects, TM1V vValues);
Definition	<pre>TM1IMPORT TM1V TM1API TM1ObjectAttributeValueSet(TM1P hPool, TM1V hObject, TM1V hAttribute, TM1V vValue);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hObjects is a valid element handle object array. hAttribute is a valid Attribute handle for the object. The possible attribute types are numeric, text, and alias, depending on the attribute you point to. vValue is the value to be assigned to the attribute. The type of value depends on the type of the attribute that you are setting.
Result	The function returns True on success, false or error on failure.
Security	The client must have WRITE rights to the object.
Errors	er_SystemParameterTypeInvalid er_ObjectSecurityNoWriteRights er_ObjectAttributeInvalidType
See Also	TM1ObjectAttributeValueSet

TM1ObjectCopy

Copies an object from one server to another.

Item	Description
Purpose	Copies an object from one server to another.
Definition	Declare Function TM1ObjectCopy Lib "tm1api.dll" (ByVal hPool As Long, ByVal hSrcObject As Long, ByVal hDstObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSrcObject is a long. This TM1 value capsule contains a handle to the object to be copied. hDstObject is a long. This TM1 value capsule contains a handle to an empty object on the destination server.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function is used during replication to copy cube and dimension data from one server to another. Typically, hSrcObject and hDstObject are on different servers. The destination object is an empty object handle of the same type as the source object. It must be an unregistered object.
Security	None.
Errors	None.
See Also	TM1ObjectDuplicate TM1CubeCreate TM1CubePerspectiveCreate TM1DimensionCreateEmpty TM1RuleCreateEmpty TM1SubsetCreateEmpty TM1ViewCreate

TM1ObjectDelete

Deletes a registered object from a server and releases its space.

Item	Description
Purpose	Deletes a registered object from a server and releases its space.
Definition	Declare Function TM1ObjectDelete Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object to be deleted.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function applies to all registered objects. The object is removed from the appropriate object list of its parent. The storage used by the object is released. All subsequent references using a handle to the object will result in the error: TM1ErrorObjectNotFound. Other errors are also possible.
Security	The client must have ADMIN privileges to the parent object.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorObjectSecurityNoAdminRights TM1ErrorObjectNotFound
See Also	TM1ObjectDestroy

TM1ObjectDestroy

Destroys an unregistered object and releases its space.

Item	Description
Purpose	Destroys an unregistered object and releases its space.

Item	Description
Definition	Declare Function TM1ObjectDestroy Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a handle to the object to be destroyed.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function applies only to major objects that are unregistered. The storage used by the object is released. Subsequent references using the handle of the object will yield unpredictable erroneous results.
Errors	TM1ErrorObjectIsRegistered TM1ErrorObjectFunctionDoesNotApply TM1ErrorObjectNotFound TM1ErrorObjectBeingUsedByObject
See Also	TM1ObjectDelete

TM1ObjectDuplicate

Makes a copy of an object in the same server or on a different server.

Item	Description
Purpose	Makes a copy of an object in the same server or on a different server.
Definition	Declare Function TM1ObjectDuplicate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object to be copied.
Result	The function returns a handle to the copy of the object.
Security	The client must have READ rights to the object to be copied.

Item	Description
Errors	TM1ErrorObjectFunctionDoesNotAp_ply TM1ErrorObjectSecurityNoReadRights
See Also	TM1ObjectCopy

TM1ObjectFileDelete

Deletes the file of a given object.

Item	Description
Purpose	Deletes the file of a given object.
Definition	Declare Function TM1ObjectFileDelete Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose file is to be deleted.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights
See Also	Other TM1ObjectFile functions.

TM1ObjectFileLoad

Used to load an object that has been unloaded.

Item	Description
Purpose	Used to load an object that has been unloaded.

Item	Description
Definition	<pre> Declare Function TM1ObjectFileLoad Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal hParent As Long, ByVal iObjectType As Long, ByVal sObjectName As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to the server on which the file resides. hParent is a long. This TM1 value capsule contains a handle to the parent of the object whose file you want to load. iObjectType is a long. This TM1 value capsule contains an object type, as defined in the module tmlapi.bas. For example, if the object is a cube, set this argument to TM1TypeCube(). If it is a dimension, set this argument to TM1TypeDimension(). For other object types, see TM1 Objects." sObjectName is a long. This TM1 value capsule contains the string name of the object.
Result	The function returns a handle to the registered object that is created when the file is loaded. The parent must be a registered object. The file to load must correspond to an object that is already registered in the server, but has been unloaded. You cannot put a file into the DB and attempt to load it.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights TM1ErrorObjectFileNotFound
See Also	Other TM1ObjectFile functions.

TM1ObjectFileSave

Saves objects after significant changes are made or new objects created.

Item	Description
Purpose	Saves objects after significant changes are made or new objects created.
Definition	<pre> Declare Function TM1ObjectFileSave Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long </pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose file is to be saved.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. Cube and Dimension objects are saved to the directory from which they came. If a previous version of the file is not found in any of the server directories, it is saved in the first one. Minor objects, such as views, are saved in the directory where their parent object resides.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights
See Also	Other TM1ObjectFile functions.

TM1ObjectListCountGet

Retrieves the number of items in a list property.

Item	Description
Purpose	Retrieves the number of items in a list property.
Definition	<pre>Declare Function TM1ObjectListCountGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal iPropertyList As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose list property is being queried. iPropertyList is a constant defined in tm1api.bas. For example, to retrieve the number of dimensions in a cube, set this variable equal to TM1CubeDimensions(). Other property index values are listed in tm1api.bas.
Result	The function returns a long integer. This long is a TM1 value capsule containing an integer. Use TM1ValIndexGet to retrieve the data. The index contains the number of items on the list. This function applies to all objects.

Item	Description
Security	The client must have READ rights to the object.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectList functions.

TM1ObjectListHandleByIndexGet

Retrieves an item on a list property given an index.

Item	Description
Purpose	Retrieves an item on a list property given an index.
Definition	<pre> Declare Function TM1ObjectListHandleByIndexGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal iPropertyList As Long, ByVal iIndex As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose list property is being queried. iPropertyList is a constant defined in tm1api.bas. For example, to retrieve a dimension handle from a cube, set this variable equal to TM1CubeDimensions(). Other property index values are listed in tm1api.bas. iIndex is a long. This TM1 value capsule contains a handle to the index of the item within the list.
Result	The function returns a handle to the requested item. This function applies to all objects.
Security	The client must have READ rights to the object.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectList functions.

TM1ObjectListHandleByNameGet

Retrieves an item in a list property given a name.

Item	Description
Purpose	Retrieves an item in a list property given a name.
Definition	<pre>Declare Function TM1ObjectListHandleByNameGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal iPropertyList As Long, ByVal sName As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose list property is being queried. iPropertyList is a constant defined in tm1api.bas. For example, to retrieve a subset from a dimension, set this variable equal to TM1DimensionSubsets(). sName is a long. This TM1 value capsule contains a handle to a string containing the name of the requested object.
Result	The function returns the handle of the requested object. This function applies to all TM1 objects except subsets. To retrieve the elements in a subset, use the function TM1ObjectListHandleByIndexGet or call TM1ObjectListHandleByNameGet passing the property TM1DimensionElements(). In summary: <pre>TM1ObjectListHandleByNameGet (hPool, hDimensionObject, TM1DimensionElements(), vsName) // This function works</pre> <pre>TM1ObjectListHandleByIndexGet (hPool, hSubsetObject, TM1SubsetElements(), vsName) // This function works</pre> <pre>TM1ObjectListHandleByNameGet (hPool, hSubsetObject, TM1SubsetElements(), vsName) // This function returns an error</pre>
Security	The client must have READ rights to the object.
Errors	<pre>TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList</pre>
See Also	Other TM1ObjectList functions.

TM1ObjectPrivateDelete

Deletes a previously registered private object.

Item	Description
Purpose	Deletes a previously registered private object.
Definition	Declare Function TM1ObjectPrivateDelete Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject a long. This TM1 value capsule contains a handle to the private object you want to delete.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The object is removed from the appropriate object list of its parent. The storage used by the object is released. All subsequent references using a handle to the object will result in the error: TM1ErrorObjectDeleted
Security	You are only allowed to delete private objects that you have created.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorObjectSecurityNoAdminRights TM1ErrorObjectDeleted
See Also	TM1ObjectPrivateRegister TM1ObjectDestroy

TM1ObjectPrivateListCountGet

Returns the number of items in the list property of a private object.

Item	Description
Purpose	Returns the number of items in the list property of a private object.
Definition	Declare Function TM1ObjectPrivateListCountGet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal iPropertyList As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose list property is being queried. It is always a parent handle. iPropertyList is a constant defined in tm1api.bas. For example, to retrieve the number of private subsets in a dimension, set this variable equal to TM1DimensionSubsets().
Result	Returns a value containing the number of items on the list.
Security	None.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectPrivate functions.

TM1ObjectPrivateListHandleByIndexGet

Returns the handle of an object.

Item	Description
Purpose	Given an index, this function returns the handle of the object in that position of a list property.
Definition	<pre> Declare Function TM1ObjectPrivateListHandleByIndexGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal iPropertyList As Long, ByVal iIndex As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose list property is being queried. iPropertyList is a constant defined in tm1api.bas. These values are returned by the object property value functions supplied by the API. iIndex is a long. This TM1 value capsule contains an index into the list indicated by the iPropertyList argument. For example, the constant TM1ObjectList returns a property index for the list property of an object. If hObject is a server handle and iPropertyList is TM1ServerDimensions(), this function returns the handle of the dimension in the iIndex position on the server.

Item	Description
Result	The function returns a handle to the requested item. This function used only to locate private sub-objects of shared objects.
Security	None.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectPrivate functions.

TM1ObjectPrivateListHandleByNameGet

Returns a handle to an object, provided that the object name is on the list.

Item	Description
Purpose	Returns a handle to an object, provided that the object name is on the list.
Definition	<pre> Declare Function TM1ObjectPrivateListHandleByNameGet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal iPropertyList As Long, ByVal sName As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object whose list property is being queried. iPropertyList is a constant defined in tmlapi.bas. These values are returned by the object property value functions supplied by the API. sName is a long. This TM1 value capsule contains a handle to a string of the name of the requested object. For example, the constant TM1ObjectList returns a property index for the list property of an object. If hObject is a server handle and iPropertyList is TM1ServerDimensions, this function returns the handle of the sName of the dimension on the server.
Result	The function returns the handle of the requested object. This function is used only to locate private sub-objects of shared objects.
Security	None.
Errors	TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyNotList
See Also	Other TM1ObjectPrivate functions.

TM1ObjectPrivatePublish

Makes a private object into a public (or shared) object.

Item	Description
Purpose	Makes a private object into a public (or shared) object.
Definition	<pre>Declare Function TM1ObjectPrivatePublish Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal sName As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object that is being published. sName is a long. This TM1 value capsule contains a handle to the name by which other applications can access the object.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function adds the name specified by sName to the list property of the parent of the object specified by hObject. This function makes a public copy of the object and assigns it a new name. All sub-objects must be public, otherwise the function will fail. The original private object is removed by this function, leaving only the new public object.
Security	To publish a private object, you must be a member of the ADMIN group.
Errors	TM1ErrorViewHasPrivateSubsets
See Also	TM1ObjectPrivateRegister

TM1ObjectPrivateRegister

Registers a private object.

Item	Description
Purpose	Registers a private object.
Definition	<pre>Declare Function TM1ObjectPrivateRegister Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hParent As Long, ByVal hObject As Long, ByVal sName As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hParent is a long. This TM1 value capsule contains a handle to the parent of the object you want to register. hObject is a long. This TM1 value capsule contains an handle to the private object you want to register. sName is a long. This TM1 value capsule contains a handle to a string which is the name under which you register the object. Applications can retrieve the object by submitting this name to the function TM1ObjectPrivateListHandleByNameGet.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function assigns a name to an object, makes it a private sub-object of its parent, and stores it permanently. Private objects can have the same name as shared objects, but this practice is not recommended.
Security	The creator of a private object has ADMIN rights to it.
Errors	None.
See Also	TM1ObjectRegister TM1ObjectPrivatePublish TM1PrivateListHandle functions

TM1ObjectPropertyGet

Retrieves the value of a property for an object.

Item	Description
Purpose	Retrieves the value of a property for an object.
Definition	<pre>Declare Function TM1ObjectPropertyGet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal vProperty As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to a valid object. vProperty is a long. This TM1 value capsule contains a handle to a property index value for the object. These values are returned by the object property value functions supplied by the API. For example, these two lines return a string containing the name of the object: <pre>vsObjectName = TM1ObjectPropertyGet(pGeneral, vhObject, TM1ObjectName()) sHierarchyName = TM1ValStringGet(hUser, vsObjectName)</pre>
Result	The function normally returns the value of the property for the object. The type of the value depends on the property, and could be any of the standard TM1 types. This function does not work for list properties. List properties must be handled using the TM1ObjectList functions. This function applies to all objects.
Security	The client must have READ access to the object in question in order to receive a result.
Errors	TM1ErrorObjectPropertyNotDefined TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectPropertyIsList

TM1ObjectPropertySet

Updates the value of a property for an object.

Item	Description
Purpose	Updates the value of a property for an object.
Definition	<pre>Declare Function TM1ObjectPropertySet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal Property_P As Long, ByVal ValRec_V As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to a valid object. Property_P is a long. This TM1 value capsule contains a handle to the property index value for the object. These values are returned by the object property value functions supplied by the API. For example, the function TM1ObjectName() returns a string containing the name of the object. ValRec_V is a long. This TM1 value capsule contains a handle to the value to be assigned to the property.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function applies to all objects. This function cannot set all properties. Some properties cannot be updated. See "Properties" for more information.
Security	The client must have WRITE rights to the object.
Errors	TM1ErrorObjectSecurityNoWriteRights TM1ErrorObjectPropertyNotDefined TM1ErrorObjectPropertyIsList
See Also	TM1ObjectPropertyGet

TM1ObjectRegister

Registers an object with its parent object.

Item	Description
Purpose	Registers an object with its parent object.
Definition	<pre> Declare Function TM1ObjectRegister Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hParent As Long, ByVal hObject As Long, ByVal sName As Long) As Long </pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hParent is a long. This TM1 value capsule contains a handle to the parent object. hObject is a long. This TM1 value capsule contains a handle to the object to be registered. sName is a long. This TM1 value capsule contains a handle to the name under which the object will be registered.
Result	Returns the handle to the registered object. The object is put in the appropriate object list of the parent object. The old handle becomes invalid.
Security	The client must have ADMIN rights to the parent object.
Errors	TM1ErrorObjectSecurityNoAdminRights TM1ErrorObjectIsRegistered TM1ErrorObjectNameInvalid TM1ErrorObjectNameIsBlank TM1ErrorObjectNameExists If the object is a Dimension, error is: TM1ErrorDimensionCannotBeCompiled If the object is a View, error is: TM1ErrorViewHasPrivateSubsets
See Also	TM1ObjectPrivateRegister

TM1ObjectReplicate

Copies an object from star server to a planet server.

Item	Description
Purpose	Copies an object from star server to a planet server.
Definition	<pre>Declare Function TM1ObjectReplicate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. hObject is a handle of the object to be replicated. This is typically a cube handle.
Result	If this function is successful, it returns a Boolean 1. The data and metadata of the requested object copied from the star server to the planet server.

Item	Description
Security	The client must have ADMIN rights to the server.
Errors	None.
See Also	TM1ObjectReplicate TM1ObjectReplicationSourceObjectName TM1CubeReplicationSyncRule TM1CubeReplicationSyncViews TM1DimensionReplicationSyncSubsets

TM1ObjectSecurityLock

Permanently prohibits WRITE access to an object.

Item	Description
Purpose	Permanently prohibits WRITE access to an object.
Definition	<pre>Declare Function TM1ObjectSecurityLock Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object to be locked.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The new restrictions take effect immediately. Only a server administrator can remove an object lock. This function applies to all objects.
Security	The client must have LOCK rights to the object.
Errors	TM1ErrorObjectSecurityNoLockRights
See Also	TM1ObjectSecurityUnLock TM1ObjectSecurityReserve

TM1ObjectSecurityRelease

Allows WRITE access to an object that was previously reserved.

Item	Description
Purpose	Allows WRITE access to an object that was previously reserved.
Definition	Declare Function TM1ObjectSecurityRelease Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object to be released.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function applies to all objects.
Security	The client must have ADMIN rights to the object, or have previously reserved the object.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityReserve

Temporarily prohibits WRITE access to an object.

Item	Description
Purpose	Temporarily prohibits WRITE access to an object.
Definition	Declare Function TM1ObjectSecurityReserve Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object to be reserved.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The new restrictions take effect immediately. This function applies to all objects.
Security	The client must have RESERVE rights to the object.
Errors	TM1ErrorObjectSecurityNoReserveRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityRightGet

Retrieves the security rights for a given object for a given group.

Item	Description
Purpose	Retrieves the security rights for a given object for a given group.
Definition	<pre> Declare Function TM1ObjectSecurityRightGet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal hGroup As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object. hGroup is a long. This TM1 value capsule contains a handle to a client or a group.
Result	The function returns an integer value indicating the current rights to the object for the group. The result will be equivalent to one of the following values: TM1SecurityRightNone() TM1SecurityRightRead() TM1SecurityRightWrite() TM1SecurityRightReserve() TM1SecurityRightLock() TM1SecurityRightAdmin() This function applies to all objects.

Item	Description
Result (cont.)	This function is designed to allow TM1 server administrators to check the access rights for clients and groups to objects on the server. If the hGroup argument is a handle to a group, the function returns the rights for the security rights for the group. If the hGroup argument is a handle to a client, the function returns the highest level of access available to the user.
Security	The client must be a member of the ADMIN group to retrieve the security for groups.
Errors	TM1ErrorObjectSecurity NoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityRightSet

Sets the security rights for a given object for a given Group.

Item	Description
Purpose	Sets the security rights for a given object for a given Group.
Definition	Declare Function TM1ObjectSecurityRightSet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long, ByVal hGroup As Long, ByVal iRight As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object. hGroup is a long. This TM1 value capsule contains a handle to the group. iRight is a long. This TM1 value capsule contains a handle to the rights level to be assigned, which is one of the following: TM1SecurityRightNone TM1SecurityRightRead TM1SecurityRightWrite TM1SecurityRightReserve TM1SecurityRightLock TM1SecurityRightAdmin This function applies to all objects.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The new rights take effect immediately.
Security	The client must be a member of the ADMIN group to set security for a group.
Errors	TM1ErrorObjectSecurity NoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ObjectSecurityUnLock

Removes a lock from an object.

Item	Description
Purpose	Removes a lock from an object.
Definition	Declare Function TM1ObjectSecurityUnLock Lib "tm1api.dll" (ByVal hPool As Long, ByVal hObject As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hObject is a long. This TM1 value capsule contains a handle to the object to be unlocked.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function applies to all objects.
Security	The client must have ADMIN rights to the object.
Errors	TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1ObjectSecurity functions.

TM1ProcessExecute

Executes a TurboIntegrator process on an IBM Cognos TM1 server.

Item	Description
Purpose	Executes a TurboIntegrator process on an IBM Cognos TM1 server.
Definition	<pre>Declare Function TM1ProcessExecute Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hProcess As Long, ByVal hParametersArray As Long) As Long</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the TM1 Server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hParametersArray is a value capsule containing an array of parameters. Each parameters can be a number (created with either TM1ValIndex or TM1ValReal functions) or a string (created with TM1ValString functions). This array has to match the exact definition of the process's parameters in number and type; if it doesn't an error is returned and the process is not executed. A process with no parameters takes an array of zero elements.
Result	The result TM1V object should first be checked to see if it an error item. If so, the error value may be retrieved. The error value may be one of the following values: TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectIsUnregistered TM1ErrorObjectInvalid If the return is not an error object is should be a Boolean object. If the Boolean is 0, the process execution generated errors. If the Boolean is 1, the operation was successful. Use the function TM1ValBoolGet to extract the Boolean from the value capsule. If you need more specific information about the error resulting from the process, call the function TM1ProcessExecuteEx in place of this function.
Security	None
Errors	As described above, the function may return one of the following error codes: TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectIsUnregistered TM1ErrorObjectInvalid The function writes all error messages to an error log file in the TM1 Server's data directory. The error log file name is the same as the process, with a time stamp appended.

Item	Description
See Also	TM1ChoreExecute

TM1ProcessExecuteEx

Executes a TurboIntegrator process on an IBM Cognos TM1 server.

Item	Description
Purpose	Executes a TurboIntegrator process on an IBM Cognos TM1 server.
Definition	<pre> Declare Function TM1ProcessExecute Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hProcess As Long, ByVal hParametersArray As Long) As Long </pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the TM1 Server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hParametersArray is a value capsule containing an array of parameters. Each parameters can be a number (created with either TM1ValIndex or TM1ValReal functions) or a string (created with TM1ValString functions). This array has to match the exact definition of the process's parameters in number and type; if it doesn't an error is returned and the process is not executed. A process with no parameters takes an array of zero elements.
Result	The result TM1V object should first be checked to see if it an error item. If so, the error value may be retrieved. The error value may be one of the following values: TM1ErrorObjectSecurityNoReadRights TM1ErrorObjectIsUnregistered TM1ErrorObjectInvalid If the return is not an error object is should contain a TM1 array. The array contains two elements. The first element is an error code. The error codes are listed below. The second element is the path to the error log file. The error log file is generated only if an error occurs. Returns an index of 1 or the object if the execution returned was normal or "process break."
Security	None

Item	Description
Errors	The returned array contains one of the following error codes. 0 (process successful) TM1ProcessAborted TM1ProcessHasMinorErrors TM1ProcessQuitCalled TM1ProcessCompletedWithMessages
See Also	TM1ChoreExecute

TM1ProcessExecuteSQLQuery

Opens a connection to an SQL da source. Builds and returns an array of records selected via the passed SQL query.

Item	Description
Purpose	Opens a connection to an SQL da source. Builds and returns an array of records selected via the passed SQL query.
Definition	<pre> Declare Function TM1ProcessExecuteSQLQuery Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hProcess As Long, ByVal hParametersArray As Long) As Long; </pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the IBM Cognos TM1 server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hParametersArray is a value capsule containing an array of parameters as follows: 1: DSN Name 2: User Name 3: Password 4: SQL Statement 5: (optional) Limit on number of records from query
Result	Returns an array of the records selected by the passed query or an error code.
Security	None

Item	Description
Errors	The returned array contains one of the following error codes. er_DatabaseInfoIncomplete er_DatabaseConnectionFailed er_DatabaseQueryExecutionFailed

TM1ProcessVariableNameIsValid

Tests whether a process variable name is valid in the specified IBM Cognos TM1 process.

Item	Description
Purpose	Tests whether a process variable name is valid in the specified IBM Cognos TM1 process.
Definition	Declare Function TM1ProcessVariableNameIsValid Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hProcess As Long, ByVal hVariableName As Long) As Long
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hProcess is a value capsule containing a valid handle to a process defined on the TM1 Server. This handle can be obtained by using the functions TM1ObjectListHandleByIndexGet or TM1ObjectListHandleByNameGet and the list property TM1ServerProcesses. hVariableName is a string value containing the process variable name.
Result	Returns an array containing bool and possibly error messages.
Security	None
Errors	The returned array contains one of the following error codes.

TM1RuleAttach

Attaches a rule to a cube.

Item	Description
Purpose	Attaches a rule to a cube.
Definition	Declare Function TM1RuleAttach Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hRule As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hRule is a long. This TM1 value capsule contains a handle to a rule.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function installs the rule as a property of its parent cube. The name of the property is TM1CubeRule.
Security	You must have ADMIN rights to the parent cube.
Errors	None.
See Also	Other TM1Rule functions.

TM1RuleCheck

Checks a rule for syntax.

Item	Description
Purpose	Checks a rule for syntax.
Definition	<pre>Declare Function TM1RuleCheck Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hRule As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hRule is a long. This TM1 value capsule contains a handle to a rule.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the syntax of the rule is correct. If the result is 0, a syntax error was detected in the rule. If the rule has a syntax error, you can retrieve the line containing the error by calling TM1ObjectPropertyGet for the rule properties TM1RuleErrorLine and TM1RuleErrorString.
Security	You must have READ access to the rule object.
Errors	None.
See Also	Other TM1Rule functions.

TM1RuleCreateEmpty

Creates an empty rule, and returns a handle to that rule.

Item	Description
Purpose	Creates an empty rule, and returns a handle to that rule.
Definition	Declare Function TM1RuleCreateEmpty Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hType As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube a long. This TM1 value capsule contains a handle to the cube to which the rule applies. hType is a long. Set this variable equal to TM1TypeRuleCalculation() for a calculation rule, and TM1TypeRuleDrill() for a drilldown rule.
Result	Returns a handle to an empty rule object. You can add lines to the rule object by calling TM1RuleLineInsert. You can compile a rule using TM1RuleCheck. Rules do not require registration, but must be attached to a cube with the function TM1RuleAttach.
Security	None.
Errors	None.
See Also	TM1RuleLineInsert TM1RuleCheck Other TM1Rule functions.

TM1RuleDetach

Detaches a rule from a cube.

Item	Description
Purpose	Detaches a rule from a cube.
Definition	Declare Function TM1RuleDetach Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hRule As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hRule is a long. This TM1 value capsule contains a handle to a rule.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function deletes the rule from the TM1CubeRule property of the parent cube.
Security	You must have ADMIN rights to the parent cube.
Errors	None.
See Also	Other TM1Rule functions.

TM1RuleLineGet

Retrieves a line from a rule.

Item	Description
Purpose	Retrieves a line from a rule.
Definition	<pre>Declare Function TM1RuleLineGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hRule As Long, ByVal iPosition As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hRule is a long. This TM1 value capsule contains a handle to a rule. iPosition is a long. This TM1 value capsule contains a value that indicates which line you want to retrieve from the rule. To retrieve the first line of the rule, set this value to 1.
Result	Returns a long. Pass the result to the function TM1ValStringGet to retrieve the line. The string contains a single line of the rule specified by hRule.
Security	You must have WRITE access to the parent cube.
Errors	None.

Item	Description
See Also	TM1RuleLineInsert TM1RuleCheck Other TM1Rule functions.

TM1RuleLineInsert

Inserts a line into a rule.

Item	Description
Purpose	Inserts a line into a rule.
Definition	<pre> Declare Function TM1RuleLineInsert Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hRule As Long, ByVal iPosition As Long, ByVal sLine As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hRule is a long. This TM1 value capsule contains a handle to a rule. A handle to a new rule is generated by the function TM1RuleCreate. iPosition is a long. This TM1 value capsule indicates the position at which the line will be inserted within the rule. To insert this line at the beginning of the rule, set this value to 1. sLine is a long. This TM1 value capsule contains a string containing the line to add to the rule.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function adds a single line of the rule specified by hRule.
Security	None.
Errors	TM1ErrorFailedToInsertLine TM1ErrorObjectNotFound
See Also	Other TM1Rule functions.

TM1ServerBatchUpdateFinish

Disables IBM Cognos TM1 batch updates.

Item	Description
Purpose	Disables IBM Cognos TM1 batch updates. Commits or discards data entered while batch update is enabled.
Definition	Declare Function TM1ServerBatchUpdateFinish Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal bDiscard As Long) As Long
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a TM1 server handle. This handle is returned by the function TM1SystemServerConnect. bDiscard is a TM1V containing a boolean. If the boolean is TRUE, all cell changes that occurred while batch update mode was enabled will be discarded. If the boolean is FALSE, all cell changes will be applied to the cubes. Calculations involving changed cubes are invalidated.
Result	This function ends batch update mode, and either applies or discards the cell changes that were made while batch update mode was enabled. Batch updates allow you to improve the performance of input-intensive applications by holding changes to cube data and saving those changes to cubes in a single batch.
Result (cont.)	When you initiate batch updates by calling TM1ServerBatchUpdateStart, IBM Cognos TM1 creates a temporary storage structure on the target server. All edits to cubes for that server are held in the temporary storage structure until you call TM1ServerBatchUpdateFinish. When you call TM1ServerBatchUpdateFinish, all edits held in temporary storage are either committed or destroyed, depending on the setting of the bDiscard flag. The temporary storage structure is destroyed. By default, batch update is disabled on a TM1 server.
Security	None.
Errors	None.
See Also	TM1ServerBatchUpdateStart

TM1ServerBatchUpdateIsActive

Returns a boolean TRUE if IBM Cognos TM1 batch update mode is enabled.

Item	Description
Purpose	Returns a boolean TRUE if IBM Cognos TM1 batch update mode is enabled.
Definition	Declare Function TM1ServerBatchUpdateIsActive Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long) As Long
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a TM1 server handle. This handle is returned by the function TM1SystemServerConnect.
Result	This function returns TRUE if batch update mode is enabled. It returns FALSE if batch update mode is disabled.
Security	None.
Errors	None.
See Also	TM1ServerBatchUpdateStart TM1ServerBatchUpdateFinish

TM1ServerBatchUpdateStart

Enables IBM Cognos TM1 batch updates.

Item	Description
Purpose	Enables IBM Cognos TM1 batch updates.
Definition	Declare Function TM1ServerBatchUpdateStart Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long) As Long
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a TM1 server handle. This handle is returned by the function TM1SystemServerConnect.

Item	Description
Result	This function enables batch update mode. Batch updates allow you to improve the performance of input-intensive applications by holding changes to cube data and saving those changes to cubes in a single batch. When you initiate batch updates, TM1 creates a temporary storage structure on the target server. All edits to cubes residing on the server are held in the temporary storage structure until you call <code>TM1ServerBatchUpdateFinish</code>. By default, batch update is disabled on a TM1 server. TM1 data spreading is disabled while batch update mode is enabled.
Security	None
Errors	None.
See Also	<code>TM1ServerBatchUpdateFinish</code>

TM1ServerLogClose

Terminates access to a server's log file.

Item	Description
Purpose	Terminates access to a server's log file.
Definition	<pre>Declare Function TM1ServerLogClose Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long) As Long</pre>
Parameters	<code>hPool</code> is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function <code>TM1ValPoolCreate</code>. <code>hServer</code> is a long. This TM1 value capsule contains a handle to a server object.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to <code>TM1ValBoolGet</code> to retrieve the call result. If the result is a 1, the operation was successful.
Security	You must have ADMIN rights to the server.
Errors	None.
See Also	Other <code>TM1ServerLog</code> functions.

TM1ServerLogNext

Retrieves the next data item from a log file.

Item	Description
Purpose	Retrieves the next data item from a log file.
Definition	Declare Function TM1ServerLogNext Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to a server object.
Result	Returns the next item (field) in the log file. The fields in a log record are as follows: Date/time of the change (string YYYYMMDDhhmmss GMT). Client performing the change (string) Transaction type (string) Old value (string or real) New value (string or real) Name of Cube changed (string) Dimension elements (from two to sixteen) (string). Boolean 0 to indicate the end of the record A Boolean 0 subsequent to the end of the last record indicates end of the log file has been reached. Note that access can be resumed later on, after more records have been written to the log, without closing and re-opening the log. A Boolean 1 indicates that the returned item is identical to the corresponding one in the previous record.
Security	You must have ADMIN rights to the server.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerLogOpen

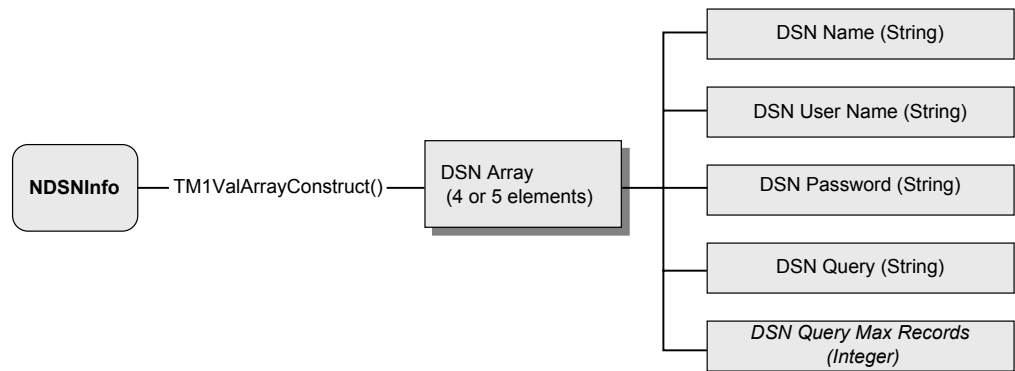
Starts access to a server's log file.

Item	Description
Purpose	Starts access to a server's log file.
Definition	<pre>Declare Function TM1ServerLogOpen Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal rStartTime As Long, ByVal sCubeFilter As long, ByVal sUserFilter As Long, ByVal sFlagFilter As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to a server object. rStartTime is a long. This TM1 value capsule contains a handle to the time stamp (GMT) after which the log records are to be retrieved. The time stamp is written as a numeric string of the form: YYYYMMDDhhmmss. sCubeFilter is a long. This TM1 value capsule contains a handle to the string pattern to match. sUserFilter is a long. This TM1 value capsule contains a string containing a TM1 client name. For example, if sUserFilter = "usr2", only log records for usr2 are written to the log file. sFlagFilter is a long. This TM1 value capsule contains a string used to filter records by flags.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function returns the first field of the first log record with a time stamp greater than rStartTime. If there are no such records, it returns a Boolean 0.
Security	You must have ADMIN rights to the TM1 server.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerOpenSQLQuery

Executes any SQL Query from the IBM Cognos TM1 client and returns a SQL Table object.

Item	Description
Purpose	Executes any SQL Query from the IBM Cognos TM1 client and returns a SQL Table object.
Definition	Declare Function TM1ServerOpenSQLQuery Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal hDsnInfo As Long) As Long
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to a server object. hDSNInfo is TM1V containing an array of 4 elements, as shown shown in the diagram following the table. The DSN referred to by DSN Name must be established on the IBM Cognos TM1 Server machine. The DSN Query Max Records element in the DSN Array is optional.
Result	The function returns a TM1V containing a TM1SqlTable object. Pass this object to the function TM1SqlGetNextRows to retrieve the data generated by the SQL statement. Typically, you follow this function call with a loop that calls TM1SqlGetNextRows until there are no more rows. Then, call TM1ObjectDestroy to destroy the SQL Query object. Unlike other TM1 objects, SQL query objects are session-dependent. You cannot save a SQL query object on the TM1 server. There is no list of SQL query objects. When you log out, all SQL Query objects are destroyed.
Security	None.
Errors	None.
See Also	TM1SQLTableGetNextRows TM1ObjectDestroy



TM1ServerPasswordChange

Changes the client's current password in a server.

Item	Description
Purpose	Changes the client's current password in a server.
Definition	Declare Function TM1ServerPasswordChange Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal sNewPassword As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hServer is a long. This TM1 value capsule contains a handle to a TM1 server. sNewPassword is a long. This TM1 value capsule contains a string to be used as the new password.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function changes the client's password.
Security	None.
Errors	None.
See Also	Other TM1ServerLog functions.

TM1ServerSecurityRefresh

Updates internal security structures with information from the IBM Cognos TM1 security cubes.

Item	Description
Purpose	Updates internal security structures with information from the IBM Cognos TM1 security cubes.
Definition	Declare Function TM1ServerSecurityRefresh Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hServer As Long);
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to the TM1 server. This handle is returned by a successful call to the function TM1SystemServerConnect.
Result	The function reads the security information from the TM1 security cubes, and updates the TM1 server's internal security information. You should call this function whenever you make one or more changes to one of the following TM1 security cubes, and you want the security changes to take effect. The TM1 security cubes are listed below. }Application_Security }Chore_Security }Client_Groups }Client_Security }Cube_Security }Dimension_Security }Process_Security
Security	None.
Errors	None.

TM1SQLTableGetNextRows

Returns rows of a SQL table object.

Item	Description
Purpose	Returns rows of a SQL table object.
Definition	Declare Function TM1SQLTableGetNextRows Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSQLTable As Long, ByVal bColumnSelection As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSQLTable is a long. This TM1 value capsule contains a handle to a SQL table object. aColumnSelection is a long. TM1 value capsule contains an array of the selected columns' name.
Result	The function returns a TM1V array which includes data of the fetched rows. Data from the specified columns is returned. You can set the number of rows by setting the SQL table object TM1SQLTableRowsetSize to the number of your choice. Set this before calling TM1ObjectPropertySet before you call TM1SqlTableGetNextRows.
Security	None.
Errors	None.
See Also	TM1CubeCellDrillListGet TM1CubeCellDrillObjectBuild

TM1SubsetAll

Populates a subset with all the elements of the parent dimension.

Item	Description
Purpose	Populates a subset with all the elements of the parent dimension.
Definition	<pre>Declare Function TM1SubsetAll Lib "tm1api.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the dimension.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function is often used to populate a subset that is to contain most of the elements in the parent dimension. After using this function, you can use TM1SubsetSelectionDelete to remove the unwanted elements from the subset.

Item	Description
Security	None.
Errors	TM1ErrorObjectNotFound
See Also	TM1SubsetCreateEmpty

TM1SubsetCreateByExpression

Creates a subset from an MDX expression.

Item	Description
Purpose	Creates a subset from an MDX expression.
Definition	<pre>Declare Function TM1API TM1SubsetCreateByExpression Lib "tm1api.dll" (ByVal hPool As Long, ByVal hServer As Long, ByVal sExpression as long);</pre>
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect. sExpression is a TM1V containing a string. The string is an MDX Expression that creates a subset.
Result	Creates a subset from an MDX expression. The expression itself can be created through the TM1 Subset Editor Record Expression command. The following shows an example of MDX generated by the subset editor. <pre>{TM1FILTERBYPATTERN({TM1SubsetBasis()}, "1*")}</pre> Since this is a dynamic subset, the subset will contain only those elements that meet the requirements of the MDX expression. The population of the subset can change over time as elements are added and removed from the dimension. Once you register the subset with TM1ObjectRegister, you can retrieve the MDX expression that created the subset by calling TM1ObjectPropertyGet, passing the property TM1SubsetExpression.
Security	None.
Errors	None.
See Also	TM1SubsetCreateEmpty TM1SubsetExpression. This is a TM1 subset property. See "Properties" for more information.

TM1SubsetCreateEmpty

Creates an empty subset object.

Item	Description
Purpose	Creates an empty subset object.
Definition	Declare Function TM1SubsetCreateEmpty Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hDim As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hDim is a long. This TM1 value capsule contains a handle to the parent dimension.
Result	Returns a handle to the subset. Subsets can be registered as public or private objects. For other TM1 clients to access the new subset, you must register the subset as a public object by calling TM1ObjectRegister. To register the subset as a private object, call TM1ObjectPrivateRegister.
Security	None.
Errors	None.
See Also	TM1ObjectRegister

TM1SubsetElementDisplay

Returns information necessary to draw levels, lines and plus/minus boxes corresponding to a subset element displayed in a tree hierarchy.

Item	Description
Purpose	Returns information necessary to draw levels, lines and plus/minus boxes corresponding to a subset element displayed in a tree hierarchy.
Definition	Declare Function TM1SubsetElementDisplay Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal iElement As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a valid handle to the subset. The handle is obtained either by calling TM1SubsetCreateEmpty or by processing a dimension list property with the TM1ObjectList functions. iElement is a long. This TM1 value capsule contains an index into the subset corresponding to the element that you want to display.
Result	Returns a long. This value capsule contains an encoded character string of information about the display characteristics of an element in a tree structure.
Result (cont.)	Once you have called this function and received the value capsule containing the string, you analyze the contents by passing the results to the following functions: TM1SubsetElementDisplayLevel TM1SubsetElementDisplayTee TM1SubsetElementDisplayEll TM1SubsetElementDisplayPlus TM1SubsetElementDisplayMinus TM1SubsetElementDisplayWeight TM1SubsetElementDisplayLine
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayEll

Returns a Boolean indicating if a subset element connector is an Ell (An Ell is the connector to the last element in a consolidation).

Item	Description
Purpose	Returns a Boolean indicating if a subset element connector is an Ell (An Ell is the connector to the last element in a consolidation).
Definition	Declare Function TM1SubsetElementDisplayEll Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vString As Long) As Integer

Item	Description
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a user handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the result is 1 (TRUE), the element is the last displayed element in a consolidation. In a tree structure, this element would be displayed with an ELL.
Security	None.
Errors	None.

TM1SubsetElementDisplayLevel

Returns a number indicating the indentation of an element in a tree structure.

Item	Description
Purpose	Returns a number indicating the indentation of an element in a tree structure.
Definition	<pre>Declare Function TM1SubsetElementDisplayLevel Lib "tm1api.dll" (ByVal hUser As Long, ByVal vString As Long) As Long</pre>
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	Returns a number indicating the indentation of the element in a tree display. For example, Year would return a display level of 0, while June would return a display level of 2.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayLine

Returns a Boolean indicating if the connector of a subset element is a line.

Item	Description
Purpose	Returns a Boolean indicating if the connector of a subset element is a line.
Definition	Declare Function TM1SubsetElementDisplayLine Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vString As Long, ByVal index As Long) As Integer
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay. index is a long integer. This TM1 value capsule contains a handle to the integer that indicates a position in the display tree from left to right. The first position is numbered 0. In the example below, the element May has a line in position 0 (the position corresponding to May's grandparent's display level), a tee in position 1, an icon in position 2, and the element name in position 3.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the element has a line preceding it in the position indicated by index.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayMinus

Returns a Boolean indicating if the element has children displayed directly beneath it in the current element list.

Item	Description
Purpose	Returns a Boolean indicating if the element has children displayed directly beneath it in the current element list.
Definition	Declare Function TM1SubsetElementDisplayMinus Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vString As Long) As Integer

Item	Description
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the result is 1 (TRUE), the element has children currently displayed directly beneath it in the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayPlus

Returns a Boolean indicating if a subset element has children that are not displayed directly beneath it in the current element list.

Item	Description
Purpose	Returns a Boolean indicating if a subset element has children that are not displayed directly beneath it in the current element list.
Definition	<pre> Declare Function TM1SubsetElementDisplayPlus Lib "tm1api.dll" (ByVal hUser As Long, ByVal vString As Long) As Integer </pre>
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the result is 1 (TRUE), the element has children that are currently not displayed directly beneath it in the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplaySelection

Returns a Boolean indicating if the subset element is currently selected

Item	Description
Purpose	Returns a Boolean indicating if the subset element is currently selected.
Definition	Declare Function TM1SubsetElementDisplaySelection Lib "tm1api.dll" (ByVal hUser As Long, ByVal vString As Long) As Long
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the result is a 1 (TRUE), the operation was successful.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayTee

Returns a Boolean indicating if the connector of a subset element is a tee.

Item	Description
Purpose	Returns a Boolean indicating if the connector of a subset element is a tee.
Definition	Declare Function TM1SubsetElementDisplayTee Lib "tm1api.dll" (ByVal hUser As Long, ByVal vString As Long) As Integer
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	If the result is a 1 (TRUE), the element is preceded with a tee connector in the display structure.
Security	None.

Item	Description
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetElementDisplayWeight

Returns the weight of an element.

Item	Description
Purpose	Returns the weight of an element.
Definition	Declare Function TM1SubsetElementDisplayWeight Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vString As Long) As Double
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a handle returned by TM1SystemOpen. vString is a long. This TM1 value capsule contains a display element string. This is the string returned by TM1SubsetElementDisplay.
Result	This function returns a double. The number is the weight of the element in the display structure. This function applies only to elements that are currently displayed as children of a parent element.
Security	None.
Errors	None.
See Also	Other TM1SubsetElementDisplay functions.

TM1SubsetInsertElement

Inserts an element into a subset.

Item	Description
Purpose	Inserts an element into a subset.
Definition	Declare Function TM1SubsetInsertElement Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal hElement As Long, ByVal iPosition As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset to which you want to add elements. hElement is a long. This TM1 value capsule contains a handle to the element you want to insert in the subset. Element handles are retrieved by calling the TM1ObjectList functions with the list property TM1SubsetElements(). iPosition is a long. This value capsule contains an integer that indicates the position into which the new element is inserted in the subset. If iPosition = 0, the element is inserted at the end of the subset. Use TM1ValIndex(hPool, 0) to set it.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. Elements can only be inserted into an unregistered subset.
Security	None. The subset into which you insert elements is unregistered. Therefore, you have exclusive access to that subset.
Errors	None.
See Also	TM1SubsetInsertSubset

TM1SubsetInsertSubset

Inserts one subset into another.

Item	Description
Purpose	Inserts one subset into another.
Definition	<pre>Declare Function TM1SubsetInsertSubset Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubsetA As Long, ByVal hSubsetB As Long, ByVal iPosition As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubsetA is a long. This TM1 value capsule contains a handle to a subset into which subset is to be inserted. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. hSubsetB is a long. This TM1 value capsule contains a handle to a subset being inserted. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. iPosition is a long. This TM1 value capsule contains an integer indicating the position to be occupied by the inserted subset. For example, if the value of the position argument is 4, the object is inserted before the fourth element of the subset. To insert an object after the last subset element, set this parameter to zero. Use TM1ValIndex(hPool, 0) to set it.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function inserts the elements of subset B into subset A. You can create subsets with repeated elements. For example, suppose you have two subsets like this: Subset 1 Subset 2 A A B Y C Z Inserting Subset 2 into Subset 1 with iPosition = TM1ValIndex(hPool, 0) yields a subset with the following elements: A, B, C, A, Y, Z Use TM1ValIndex(hPool, 0) to set iPosition.
Security	If the subset is a public object, you must have WRITE access to the dimension containing the subset. If the subset is unregistered or private, no security restrictions apply.
Errors	None.
See Also	TM1SubsetInsertElement

TM1SubsetSelectByAttribute

Selects elements of a subset that have an attribute matching value.

Item	Description
Purpose	Selects elements of a subset that have an attribute matching value.
Definition	<pre>Declare Function TM1SubsetSelectByAttribute Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal hAlias As Long, ByVal sValueToMatch As Long, ByVal bSelection As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset. It is obtained either by calling TM1SubsetCreateEmpty, or by processing a dimension list property with the TM1ObjectList functions. hAlias is a long. This TM1 value capsule contains a handle to an attribute. This handle is obtained by calling one of the TM1ObjectListHandle functions, and specifying the property TM1ObjectAttributes(). This call must be made on the parent dimension of the subset, not on the subset itself. sValueToMatch is a long. This TM1 value capsule contains a string or numeric value of an attribute. bSelection is a long. This TM1 value capsule contains a Boolean. If the Boolean is TRUE, the element corresponding to the index is selected. If the Boolean is FALSE, the element corresponding to the element is de-selected.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. Selects elements in a subset that have a specified attribute (as indicated by hAttr) set to a specified value (as indicated by sValueToMatch).
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectByIndex

Selects an element of a subset by its index.

Item	Description
Purpose	Selects an element of a subset by its index.
Definition	Declare Function TM1SubsetSelectbyIndex Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal iPosition As Long, ByVal bSelection As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. iPosition is a long. This TM1 value capsule contains a handle to an integer containing the position of the element to select. bSelection is a long. This TM1 value capsule contains a Boolean. If the Boolean is TRUE (1), the element corresponding to the index is selected. If the Boolean is FALSE (0), the element corresponding to the element is de-selected.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectByLevel

Selects or de-selects all elements with a given level.

Item	Description
Purpose	Selects or de-selects all elements with a given level.
Definition	Declare Function TM1SubsetSelectByLevel Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal iLevel As Long, ByVal bSelection As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. iLevel is a long. This TM1 value capsule contains a handle to an integer indicating the level of element to select or de-select. bSelection is a long. This TM1 value capsule contains a Boolean. If this value is TRUE (1), all the elements of the specified level are selected. If the value is FALSE (0), all the elements of the given level are de-selected.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectByPattern

Selects all elements whose names match a given regular expression pattern.

Item	Description
Purpose	Selects all elements whose names match a given regular expression pattern.
Definition	<pre>Declare Function TM1SubsetSelectByPattern Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal sPattern As Long, ByVal hElement As Long) As Long</pre>

Item	Description
Parameters	<i>hPool</i> is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function <code>TM1ValPoolCreate</code>. <i>hSubset</i> is a long. This TM1 value capsule contains a handle to the subset from which you want to select elements. The handle is obtained with <code>TM1SubsetCreateEmpty</code> or one of the <code>TM1ListHandle</code> functions. <i>sPattern</i> is a long. This TM1 value capsule contains a string pattern. The pattern can contain wild card characters, such as * and ?. If the search is not for an exact match, you must use the *. For example, a search for "bird" will not find birds. A search for "bird*" will find birds. A search for "b*" will find birds. And a search for "birds" will find birds. <i>hElement</i> is a long. This TM1 value capsule contains a handle to a Boolean. If the Boolean is TRUE, elements matching the pattern are selected. If the Boolean is FALSE, elements matching the pattern are de-selected.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to <code>TM1ValBoolGet</code> to retrieve the call result. If the result is a 1, the operation was successful. Elements matching the pattern are either selected or de-selected. The pattern matching is applied to raw element names. Aliases applied to elements in the subset are not examined by this function.
Security	None.
Errors	None.
See Also	<code>TM1SubsetSelectByIndex</code>

TM1SubsetSelectionDelete

Deletes selected elements from a subset.

Item	Description
Purpose	Deletes selected elements from a subset.
Definition	<pre> Declare Function TM1SubsetSelectionDelete Lib "tm1api.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long </pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset from which you want to select elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. All elements that were previously selected through one or more of the TM1SubsetSelect functions are now deleted from the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSelectionInsertChildren

Takes each selected element and inserts its children, if any, directly under the element in the list.

Item	Description
Purpose	Takes each selected element and inserts its children, if any, directly under the element in the list. This function is used to drill down on the elements in a subset.
Definition	<pre>Declare Function TM1SubsetSelectionInsertChildren Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset into which you want to insert elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The subset referenced by the handle now contains the children of the elements that were previously selected in the subset. If the children are already present, this function inserts them again. It is the application's responsibility to check for and eliminate duplicates.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions. TM1SubsetSelectionInsertParents

TM1SubsetSelectionInsertParents

Inserts the parents of each selected element directly above the element in the list.

Item	Description
Purpose	Inserts the parents of each selected element directly above the element in the list.
Definition	<pre> Declare Function TM1SubsetSelectionInsertParents Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset into which you want to insert elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	Inserts the parents of each selected element above the element. If an element is a member of more than one consolidation, all of its parents are inserted into the list. If the parents are already present, this function inserts them again. It is the application's responsibility to check for and eliminate duplicates.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1SubsetSelect functions. TM1SubsetSelectionInsertChildren

TM1SubsetSelectionKeep

Removes all elements from the subset that are not selected.

Item	Description
Purpose	Removes all elements from the subset that are not selected.
Definition	Declare Function TM1SubsetSelectionKeep Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function removes all elements from a subset that are not selected by one of the TM1SubsetSelect functions.
Security	None.
Errors	None.
See Also	TM1SubsetCreateEmpty

TM1SubsetSelectNone

Clears the selection flag from any selected elements in a subset.

Item	Description
Purpose	Clears the selection flag from any selected elements in a subset.
Definition	Declare Function TM1SubsetSelectNone Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset into which you want to insert elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	Clears the selection flag for all elements in the subset.
Security	None.
Errors	None.
See Also	Other TM1SubsetSelect functions.

TM1SubsetSort

Sorts the elements in a subset alphabetically.

Item	Description
Purpose	Sorts the elements in a subset alphabetically.
Definition	<pre>Declare Function TM1SubsetSort Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubset As Long, ByVal bSortDown As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset into which you want to insert elements. bSortDown is a long. This TM1 value capsule contains a Boolean. If the Boolean is FALSE (0), the elements in the subset are sorted in alphabetical order from A to Z. If the Boolean is TRUE (1), the subset elements are sorted in reverse alphabetical order from Z to A.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is 1, the subset elements are sorted from Z through A. If the result is 0, the subset elements are sorted from A through Z. Use the parameter bSortDown to set the sorting order. The sorting is applied to raw element names. This function does not examine aliases applied to elements in the subset.
Security	None.

Item	Description
Errors	None.
See Also	Other TM1Subset functions.

TM1SubsetSortByHierarchy

Sorts the elements of a subset according to their parent / child relationships.

Item	Description
Purpose	Sorts the elements of a subset according to their parent / child relationships.
Definition	Declare Function TM1SubsetSortByHierarchy Lib "tm1api.dll" (ByVal hPool As Long, ByVal hSubset As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubset is a long. This TM1 value capsule contains a handle to the subset that you want to sort.
Result	This function sorts the subset as follows: All the elements that have neither parents nor children are grouped first, in alphabetical order. All the consolidated elements containing at least one child element are sorted in alphabetical order. All the child objects are grouped below their parents, and are sorted in alphabetical order.
Security	None.
Errors	None.
See also	TM1SubsetSort

TM1SubsetSubtract

Removes a set of elements from a subset.

Item	Description
Purpose	Removes a set of elements from a subset.

Item	Description
Definition	<pre> Declare Function TM1SubsetSubtract Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hSubsetA As Long, ByVal hSubsetB As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hSubsetA is a long. This TM1 value capsule contains a handle to the subset from which you want to delete elements. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions. hSubsetB is a long. This TM1 value capsule contains a handle to the subset whose member elements you want to delete from Subset A. The handle is obtained with TM1SubsetCreateEmpty or one of the TM1ListHandle functions.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function eliminates from Subset A any elements that are common to both Subset A and Subset B.
Security	None.
Errors	None.
See Also	TM1SubsetInsertSubset

TM1SubsetUpdate

Replaces a registered subset with a new one.

Item	Description
Purpose	Replaces a registered subset with a new one.
Definition	<pre> Declare Function TM1SubsetUpdate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hOldSubset As Long, ByVal hNewSubset As Long) As Long </pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hOldSubset is a long. This TM1 value capsule contains a handle to the registered subset to be replaced. hNewSubset is a long. This TM1 value capsule contains a handle to the subset that replaces the old one.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The old subset is destroyed and replaced with the new one. All affected views are updated accordingly.
Security	The client must have ADMIN rights to the dimension being updated.
Errors	TM1ErrorObjectIsUnregistered TM1ErrorObjectSecurityNoAdminRights
See Also	Other TM1Subset functions.

TM1SystemAdminHostGet_VB

Retrieves the name of the AdminHost server.

Item	Description
Purpose	Retrieves the name of the AdminHost server.
Definition	<pre>Declare Sub TM1SystemAdminHostGet_VB Lib "tmlapi.dll" (ByVal hUser As Long, ByVal AdminHost As String, ByVal max As Integer)</pre>
Parameters	hUser is a long. It is a valid user handle obtained with TM1SystemOpen. AdminHost is an empty string. Max is the length of the AdminHost string.
Result	This subroutine returns the name of the IBM Cognos TM1 ADMIN host.
Security	None.
Errors	None.
See Also	TM1SystemAdminHostSet

TM1SystemAdminHostSet

Sets the name of the AdminHost server.

Item	Description
Purpose	Sets the name of the AdminHost server.
Definition	<pre>Declare Sub TM1SystemAdminHostSet Lib "tmlapi.dll" (ByVal hUser As Long, ByVal AdminHosts As String)</pre>
Parameters	hUser is a long. It is a valid user handle obtained with TM1SystemOpen. AdminHosts is a NULL-terminated string specifying a list of host names separated by commas.
Result	This function must be called before any TM1SystemServer functions and may be called at any time to reset the list of available servers. This function does not affect existing connections.
Security	None.
Errors	None.
See Also	TM1SystemOpen

TM1SystemBuildNumber_VB

Returns a string corresponding to the build number of the IBM Cognos TM1 Server.

Item	Description
Purpose	Returns a string corresponding to the build number of the IBM Cognos TM1 Server.
Definition	<pre>Declare Sub TM1SystemBuildNumber_VB Lib "tmlapi.dll" (ByVal str As String, ByVal Max As Integer)</pre>
Parameters	Str is a string. The string should be empty when you make the function call. When the call successfully completes, the string contains the build number of the TM1 Server. Max is an integer. This is the maximum length of the string that can be accepted in the str variable. For example, suppose you declare str as 100 characters: <pre>dim str as string * 100</pre> You should set Max equal to 100.

Item	Description
Result	The function returns a build string like the following: 8.2.2.1209
Security	None.
Errors	None.
See Also	TM1SystemOpen

TM1SystemClose

Disconnects the user from the API and releases resources.

Item	Description
Purpose	Disconnects the user from the API and releases resources.
Definition	Declare Sub TM1SystemClose Lib "tm1api.dll" (ByVal hUser As Long)
Parameters	hUser is a long. It is a handle obtained with TM1SystemOpen.
Results	Before you can disconnect from the server, you must run TM1SystemServerDisconnect(). Then, when you run TM1SystemClose(), the user is disconnected from the API and all resources are released. Any existing connections are closed.
Security	None.
Errors	None.
See Also	TM1SystemOpen

TM1SystemOpen

Connects the user to the API.

Item	Description
Purpose	Connects the user to the API.
Definition	Declare Function TM1SystemOpen Lib "tm1api.dll" () As Long
Parameters	None.

Item	Description
Result	The function returns a user handle. Typically, this user handle is used to create a pool handle with TM1ValPoolCreate. The pool handle is then passed to other API calls as an argument. This function is part of the API initialization sequence required by every TM1 API program. For more information, see “Connecting to the API” on page 5.
Security	None.
Errors	None.
See Also	TM1SystemClose

TM1SystemServerClientName_VB

Returns a client's name.

Item	Description
Purpose	Returns a client's name.
Definition	<pre>Declare Sub TM1SystemServerClientName_VB Lib "tmlapi.dll" (ByVal hUser As Long, ByVal index As Integer, ByVal res As String, ByVal max As Integer)</pre>
Parameters	hUser is a long. It is user handle returned by the function TM1SystemServerConnect. index is an integer. This integer is an offset into the list of available servers currently available to the client. These servers are listed in the ADMIN server user interface. res is a string where the result is returned. max is an integer indicating the maximum length of a string that can be accepted by res.
Result	The function returns a string. The string contains the name of the current user. Use TM1ValStringGet to retrieve information.
Security	None.
Errors	None.
See Also	TM1SystemServerConnect

TM1SystemServerConnect

Connects a client to a server.

Item	Description
Purpose	Connects a client to a server. Use this function to start a TM1 session if your IBM Cognos TM1 Server is configured for standard TM1 authentication or LDAP authentication. Use TM1SystemServerConnectIntegratedLogin if it is configured for Integrated Login.
Definition	Declare Function TM1SystemServerConnect Lib "tmlapi.dll" (ByVal hPool As Long, ByVal sDatabase As Long, ByVal sClient As Long, ByVal sPassword As Long) As Long
Parameters	hPool is a long. This TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. sDatabase is a long. This TM1 value capsule contains a string value containing the name of the server. sClient is a long. This TM1 value capsule contains a string value containing the name of the client. sPassword is a long. This TM1 value capsule contains a string value containing the password.
Result	The function returns a handle to the server. This function is part of the API initialization sequence required by every TM1 API program. For more information, see “Connecting to the API” on page 5.
Errors	TM1ErrorSystemServerNotFound TM1ErrorSystemServerConnectionFailed TM1ErrorSystemServerClientAlready Connected TM1ErrorSystemServerClientNotFound TM1ErrorSystemServerClientPasswordInvalid TM1ErrorSystemServerClientPasswordExpired TM1ErrorSystemServerClientExceedMax LogonNumber TM1ErrorClientMaximumPortsExceeded
See Also	TM1SystemOpen TM1SystemServerDisconnect

TM1SystemServerConnectIntegratedLogin

Connects a client to a server using Integrated Login.

Item	Description
Purpose	Connects a client to a server using Integrated Login. Integrated Login allows you to use your Windows security system to authenticate IBM Cognos TM1 users. For more information on integrated login, see the IBM Cognos TM1 <i>Operations Guide</i> . Use the function TM1SystemServerConnect instead of this function if your TM1 server is configured for standard TM1 authentication or LDAP authentication.
Definition	<pre>Declare Function TM1SystemServerConnectIntegratedLogin Lib "tmlapi.dll" (ByVal hPool As Long, ByVal sDatabase As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. sDatabase is a long. This TM1 value capsule contains a string value containing the name of the server.
Result	The function returns a handle to the server. This function attempts to connect to the server through Integrated Login. The login is attempted using the windows domain name under which the user logged in to the network. for example, suppose you follow this sequence: You log in to a windows workstation as Stewart in the Germany domain. You run an application that includes Tm1SystemServerConnectIntegratedLogin. The API will try to log in to TM1 using the id Stewart. The TM1 server must be configured to accept Integrated Logins. (The IntegratedSecurityMode parameter in tm1s.cfg must be set to 2 or 3.)
Errors	TM1ErrorSystemServerIntegrated SecurityRefused TM1ErrorSystemServerIntegrated SecurityRequired
See Also	TM1SystemOpen TM1SystemServerConnect TM1SystemServerDisconnect

TM1SystemServerDisconnect

Disconnects a user from a server.

Item	Description
Purpose	Disconnects a user from a server.
Definition	Declare Function TM1SystemServerDisconnect Lib "tm1api.dll" (ByVal hPool As Long, ByVal hDatabase As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hDatabase is a long. This TM1 value capsule contains a string value containing the name of the server.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function is part of the cleanup and logout sequence required for a well-behaved TM1 API program. For more information, see “Disconnecting from the API” on page 8.
Security	None.
Errors	TM1ErrorSystemServerClientNotConnected
See Also	Other TM1SystemServer functions.

TM1SystemServerHandle

Returns the handle to a server given its name.

Item	Description
Purpose	Returns the handle to a server given its name. This function can only returns a handle for a server to which you have already established a connection.
Definition	Declare Function TM1SystemServerHandle Lib "tm1api.dll" (ByVal hUser As Long, ByVal name As String) As Long
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login. name is a string value containing the name of the server.

Item	Description
Result	The function returns a long integer. The long is a value capsule. The value capsule contains the handle to the server.
Security	None.
Errors	If the function fails, it returns a long. This IBM Cognos TM1 value capsule contains a Boolean 0. Use the function TM1ValBoolGet to extract the Boolean from the value capsule.
See Also	Other TM1SystemServer functions.

TM1SystemServerName_VB

Returns the name of a server in the list of available servers given an index as a Microsoft Visual Basic string.

Item	Description
Purpose	Returns the name of a server in the list of available servers given an index as a Microsoft Visual Basic string.
Definition	<pre> Declare Sub TM1SystemServerName_VB (ByVal hUser As Long, ByVal index As Integer, ByVal Retval String, ByVal Max As Integer) </pre>
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login. index indicates the position of the server in the list. Retval is a string where the result is returned. Max is the maximum length of string that can be accepted by Retval. Max must be numeric.
Result	The string Retval is returned padded with blanks. It must be declared with a fixed maximum length. The length should match the value of Max. For example: <pre> Dim Retval as String * 75 TM1SystemServerName_VB(hUser, index, Retval, 75) </pre>
Security	None.
Errors	None.

TM1SystemServerNof

Returns the number of available servers.

Item	Description
Purpose	Returns the number of available servers.
Definition	Declare Function TM1SystemServerNof Lib "tmlapi.dll" (ByVal hUser As Long) As Integer
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login.
Result	The function returns the number of available servers. You should call TM1SystemServerReload before calling this function.
Security	None.
Errors	None.
See Also	TM1SystemServerReload

TM1SystemServerReload

Loads information from the IBM Cognos TM1 ADMIN server into the API.

Item	Description
Purpose	Loads information from the IBM Cognos TM1 ADMIN server into the API.
Definition	Declare Sub TM1SystemServerReload Lib "tmlapi.dll" (ByVal hUser As Long)
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login.
Result	The function loads information from the TM1 ADMIN Server into the API. In order to get an accurate count of available servers, you must call TM1SystemServerReload, then call TM1SystemServerNof.
Security	None.
Errors	None.
See Also	TM1SystemServerNof

TM1SystemServerStart

Starts the TM1 server.

Item	Description
Purpose	Starts the TM1 server.
Definition	<pre>Declare Function TM1SystemServerStart Lib "tmlapi.dll" (ByVal hUser As Long, ByVal szName As String, ByVal szDataDirectory As String, ByVal szAdminHost As String, ByVal szProtocol As String, ByVal iPortNumber As Integer) As Integer</pre>
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a user handle as returned by Integrated Login. szName is a null-terminated string containing the name of the server to start. szDataDirectory is a null-terminated string containing the path of the TM1 data directory. szAdminHost is a null-terminated string containing the path of the TM1 ADMIN directory. szProtocol is a null-terminated string containing the protocol to use to connect to the server. For TCP/IP, you should use the string "tcp." For IPX, you should use the string "ipx." iPortNumber is an integer containing the port number of the server. The default port number for the TM1 server is 5000.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function allows you to start a server on the local machine only, not on other machines in the network. This function does not work on a TM1 server running on UNIX.
Security	None.
Errors	None.
See Also	TM1SystemServerStop

TM1SystemServerStop

Stops an IBM Cognos TM1 server.

Item	Description
Purpose	Stops an IBM Cognos TM1 server.

Item	Description
Definition	Declare Function TM1SystemServerStop Lib "tmlapi.dll" (ByVal hUser As Long, ByVal szName As String, ByVal bSave As Integer) As Integer
Parameters	hUser is a long. It is a user handle obtained with Integrated Login. szName is a null-terminated string containing the name of the server to stop. bSave is a Boolean. If the Boolean is TRUE (1), changes to TM1 server data in memory are written to the hard disk before the server is brought down. If the Boolean is FALSE (0), no changes are written to disk before the server is brought down.
Result	Returns an integer. If the result is a 1, the operation was successful.
Security	None.
Errors	None.
See Also	TM1SystemServerStart

TM1SystemVersionGet

Returns the current version of the API.

Item	Description
Purpose	Returns the current version of the API.
Definition	Declare Function TM1SystemVersionGet Lib "tmlapi.dll" () As Integer
Parameters	None.
Result	Returns an integer indicating the version of the API multiplied by 100. For example version 2.5 will result in 250.
Security	None.
Errors	None.

TM1ValArrayGet

Retrieves a component of an array value.

Item	Description
Purpose	Retrieves a component of an array value.

Item	Description
Definition	<pre> Declare Function TM1ValArrayGet Lib "tm1api.dll" (ByVal hUser As Long, ByVal vArray As Long, ByVal index As integer) As Long </pre>
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login. vArray is a long. This IBM Cognos TM1 value capsule contains a handle to an array value. index is an integer. This integer is a one-based position within the array.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function returns the value handle stored at the position given by index.
Security	None.
Errors	The function returns a zero if errors are encountered.
See Also	Other TM1ValArray functions.

TM1ValArrayGet

Retrieves a component of an array value.

Item	Description
Purpose	Retrieves a component of an array value.
Definition	<pre> Declare Function TM1ValArrayGet Lib "tm1api.dll" (ByVal hUser As Long, ByVal vArray As Long, ByVal index As integer) As Long </pre>
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login. vArray is a long. This IBM Cognos TM1 value capsule contains a handle to an array value. index is an integer. This integer is a one-based position within the array.

Item	Description
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. The function returns the value handle stored at the position given by index.
Security	None.
Errors	The function returns a zero if errors are encountered.
See Also	Other TM1ValArray functions.

TM1ValArrayMaxSize

Returns the largest number of components an array value can hold.

Item	Description
Purpose	Returns the largest number of components an array value can hold.
Definition	Declare Function TM1ValArrayMaxSize Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vArray As Long) As Long
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login. vArray is a long. This IBM Cognos TM1 value capsule contains a handle to an array value.
Result	The function returns the maximum number of array values that vArray can contain. The function returns zero if an error occurs.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValArraySet

Updates a component of an array value.

Item	Description
Purpose	Updates a component of an array value.

Item	Description
Definition	<pre> Declare Sub TM1ValArraySet Lib "tm1api.dll" (ByVal vArray As Long, ByVal val As Long, ByVal index As Long) </pre>
Parameters	vArray is a long. This IBM Cognos TM1 value capsule contains a handle to a TM1 value array. val is a long. This TM1 value capsule contains the value handle to the value to be stored in the array. index is a long. This long integer is a one-based position within the array.
Result	The current value handle at position Index within array val is replaced by newval. Note that overwriting an object handle in an array does not destroy the underlying object on the TM1 server.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValArraySetSize

Establishes an array value of a given size.

Item	Description
Purpose	Establishes an array value of a given size.
Definition	<pre> Declare Sub TM1ValArraySetSize Lib "tm1api.dll" (ByVal vArray As Long, ByVal Size As Long) </pre>
Parameters	vArray is a long. This IBM Cognos TM1 value capsule contains a handle to an array value. Size is a long. This TM1 value capsule contains a handle to an index indicating the size of the array you are creating.
Result	Establishes the value capsule as an array of Size elements. Call TM1ValArray before you call this function.
Security	None.
Errors	None.
See Also	Other TM1ValArray functions.

TM1ValBool

Constructs a Boolean value capsule.

Item	Description
Purpose	Constructs a Boolean value capsule.
Definition	Declare Function TM1ValBool Lib "tmlapi.dll" (ByVal hPool As Long, ByVal InitBool As Integer) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. InitBool is the value to be stored in the value capsule.
Result	The function returns the handle to the value capsule created.
Security	None.
Errors	If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
See Also	Other TM1ValBool functions.

TM1ValBoolGet

Retrieves the contents of a Boolean value capsule.

Item	Description
Purpose	Retrieves the contents of a Boolean value capsule.
Definition	Declare Function TM1ValBoolGet Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vBool As Long) As Integer
Parameters	hUser is a long. It is a handle obtained with Integrated Login. vBool is a long. This IBM Cognos TM1 value capsule contains a Boolean.
Result	The function returns the Boolean content of the value capsule.
Security	None.
Errors	If there is an error, the function returns zero.
See Also	Other TM1ValBool functions.

TM1ValBoolSet

Update the contents of a Boolean value capsule.

Item	Description
Purpose	Update the contents of a Boolean value capsule.
Definition	<pre>Declare Sub TM1ValBoolSet Lib "tmlapi.dll" (ByVal vBool As Long, ByVal Bool As Long)</pre>
Parameters	vBool is a long. This IBM Cognos TM1 value capsule contains a handle to the Boolean value capsule whose contents is to be updated. Bool is a long. This TM1 value capsule contains a handle to the value used to update the capsule.
Result	The function updates the target Boolean value capsule with the new value.
Security	None.
Errors	None.
See Also	Other TM1ValBool functions.

TM1ValErrorCode

Extracts the error code from an error value.

Item	Description
Purpose	Extracts the error code from an error value.
Definition	<pre>Declare Function TM1ValErrorCode Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vError As Long) As Long</pre>
Parameters	hUser is a long. It is a handle obtained with Integrated Login. vError is a long. This IBM Cognos TM1 value capsule contains a handle to the error whose contents is to be retrieved.
Result	The function returns the error code. You can pass this error code to TM1ValErrorString to receive an error message string.
Security	None.
Errors	If an error occurs, the function returns zero.
See Also	TM1ValErrorString_VB

TM1ValErrorString_VB

Retrieves a Microsoft Visual Basic string corresponding to an IBM Cognos TM1 error code.

Item	Description
Purpose	Retrieves a Microsoft Visual Basic string corresponding to an IBM Cognos TM1 error code.
Definition	<pre>Declare Sub TM1ValErrorString_VB(ByVal hUser as Long, ByVal vValue as Long, ByVal Str as String, ByVal Max as Integer)</pre>
Parameters	hUser is a long. It is a handle obtained with Integrated Login. vValue is a long. It is a TM1 value capsule contains a TM1 error number. Str is a string where the result is returned. Max is the maximum length of string that can be accepted by Str. Max must be numeric
Result	Returns the string Str padded with blanks. It must be declared with a fixed maximum length. The length should match the value of Max. For example: <pre>Dim Str as String * 75 TM1ValStringGet_VB(hUser, vValue, Str, 75)</pre>
Security	None.
Errors	None.
See Also	TM1ValErrorCode

TM1ValIndex

To construct a value capsule containing an index (32-bit integer).

Item	Description
Purpose	To construct a value capsule containing an index (32-bit integer).
Definition	<pre>Declare Function TM1ValIndex Lib "tmlapi.dll" (ByVal hPool As Long, ByVal InitIndex As Long) As Long</pre>

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. InitIndex is a long. This TM1 value capsule contains the value to be stored in the capsule.
Result	The function returns the handle to the value capsule created.
Security	None.
Errors	If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
See Also	TM1ValIndexGet TM1ValIndexSet

TM1ValIndexGet

Retrieves the contents of an index value capsule.

Item	Description
Purpose	Retrieves the contents of an index value capsule.
Definition	<pre>Declare Function TM1ValIndexGet Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vIndex As Long) As Long</pre>
Parameters	hUser is a long. It is a valid user handle obtained with Integrated Login. vIndex is a long. This is the value capsule whose contents is to be retrieved.
Result	The function returns a long integer. This long is a handle to an IBM Cognos TM1 value capsule containing an integer. If the function returns 0, the operation was not successful. The function returns the index contents of the value capsule.
Security	None.
Errors	None.
See Also	Other TM1ValIndex functions.

TM1ValIndexSet

Update the contents of an index value capsule.

Item	Description
Purpose	Update the contents of an index value capsule.
Definition	Declare Sub TM1ValIndexSet Lib "tmlapi.dll" (ByVal vIndex As Long, ByVal index As Long)
Parameters	vIndex is a long. This is the value capsule whose contents is to be updated. index is a long. This IBM Cognos TM1 value capsule contains the value used to update the capsule.
Result	The function updates the target index value capsule with the new value.
Security	None.
Errors	None.
See Also	Other TM1ValIndex functions.

TM1ValsUndefined

Tests whether a value is of type TM1CubeCellValueUndefined().

Item	Description
Purpose	Tests whether a value is of type TM1CubeCellValueUndefined().
Definition	Declare Function TM1ValIsUndefined Lib "tmlapi.dll" (ByVal hUser As Long, ByVal Value As Long) As Long
Parameters	hUser is a TM1Val. This IBM Cognos TM1 value capsule contains a valid user handle. This handle can be obtained with Integrated Login. Value is the TM1 value capsule to be tested.
Result	Returns a TM1 value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the value is of type TM1CubeCellValueUndefined. Otherwise, the function returns 0.
Security	None.
Errors	None.

TM1ValsUpdatable

Tests whether a value retrieved from a server can be updated.

Item	Description
Purpose	Tests whether a value retrieved from a server can be updated.
Definition	Declare Function TM1ValIsUpdatable Lib "tmlapi.dll" (ByVal hUser As Long, ByVal Value As Long) As Integer
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. Value is a long. It is the IBM Cognos TM1 value capsule to be tested.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the operation was successful. This function applies to cell values and object properties.
Security	None.
Errors	None.
See Also	TM1ValIsUndefined

TM1ValObject

To construct a value capsule containing an object handle.

Item	Description
Purpose	To construct a value capsule containing an object handle.
Definition	Declare Function TM1ValObject Lib "tmlapi.dll" (ByVal hPool As Long, ByRef InitObject As String) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. InitObject is an object handle to be stored in the value capsule.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.

Item	Description
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectCanRead

Determines whether the client has READ access to an object.

Item	Description
Purpose	Determines whether the client has READ access to an object.
Definition	Declare Function TM1ValObjectCanRead Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vObject As Long) As Integer
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a valid user handle obtained with Integrated Login. vObject is a long. This TM1 value capsule contains the object handle.
Result	Returns a Boolean value. If the result is a 1, one of the groups to which the client belongs has READ or higher rights to the object. Otherwise, it returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectCanWrite

Determines whether the client has WRITE access to an object.

Item	Description
Purpose	Determines whether the client has WRITE access to an object.
Definition	Declare Function TM1ValObjectCanWrite Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vObject As Long) As Integer
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. vObject is a long. This is the value containing the object handle.

Item	Description
Result	Returns a Boolean value. If the result is a 1, one of the groups to which the client belongs has WRITE or higher rights to the object, provided that the object is not reserved or locked. Otherwise, it returns zero. If there is an error, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectGet

Retrieves the contents of an object value capsule.

Item	Description
Purpose	Retrieves the contents of an object value capsule.
Definition	<pre> Declare Sub TM1ValObjectGet Lib "tm1api.dll" (ByVal hUser As Long, ByVal vObject As Long, ByVal pObject As String) </pre>
Parameters	hUser is a long. This IBM Cognos TM1 value capsule contains a valid user handle obtained with Integrated Login. vObject is a long. This TM1 value capsule contains a handle to the value capsule whose contents is to be retrieved. pObject is an empty string.
Result	This function returns nothing when successful. pObject is a TM1 object handle that was extracted from the value capsule.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectSet

Update the contents of an object value capsule.

Item	Description
Purpose	Update the contents of an object value capsule.
Definition	Declare Sub TM1ValObjectSet Lib "tmlapi.dll" (ByVal vObject As Long, ByVal pObject As String)
Parameters	vObject is a long. This IBM Cognos TM1 value capsule contains the object handle that is to be updated. pObject is a long. This TM1 object is used to update the capsule.
Result	The function updates the target object value capsule with the new value.
Security	None.
Errors	None.
See Also	Other TM1ValObject functions.

TM1ValObjectType

Retrieves the type of object.

Item	Description
Purpose	Retrieves the type of object.
Definition	Declare Function TM1ValObjectType Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vObject As Long) As Long
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. vObject is a long. This IBM Cognos TM1 value capsule contains a TM1 object handle.
Result	Returns an integer. The integer is one of the TM1ValType constants defined in tmlapi.bas. For example, if the object is a cube, TM1ObjectType returns the constant Tm1TypeCube(). The object types are defined in tmlapi.bas.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1ValObject functions.

TM1ValPoolCount

Returns the number of values stored in a value pool.

Item	Description
Purpose	Returns the number of values stored in a value pool.
Definition	<pre> Declare Function TM1ValPoolCount Lib "tmlapi.dll" (ByVal hPool As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate.
Result	The function returns a long integer. This long is a handle to a TM1 value capsule containing an integer. The value of the integer indicates the number of values in the pool. If the function returns 0, the operation was not successful.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolCreate

Creates a new value pool.

Item	Description
Purpose	Creates a new value pool.
Definition	<pre> Declare Function TM1ValPoolCreate Lib "tmlapi.dll" (ByVal hUser As Long) As Long </pre>
Parameters	hUser is a long. This is a user handle obtained with Integrated Login.
Result	The function returns a handle to the pool. The handle is valid until the TM1ValPoolDestroy function is called. If there is an error, the function returns zero.
Security	None.
Errors	None.

Item	Description
See Also	Other TM1ValPool functions.

TM1ValPoolDestroy

Clears a value pool.

Item	Description
Purpose	Clears a value pool.
Definition	Declare Sub TM1ValPoolDestroy Lib "tmlapi.dll" (ByVal hPool As Long)
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate.
Result	This function does not return a value. The value pool is cleared. Any value handles referring to the value pool become invalid. Using such handles will cause unpredictable results. The memory occupied by the value pool is retained by the TM1 server. It is not released back to the operating system. This function is part of the cleanup and logout sequence required for a well-behaved TM1 API program. For more information, see "Disconnecting from the API" on page 8.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolGet

Retrieves a value from a value pool.

Item	Description
Purpose	Retrieves a value from a value pool.
Definition	Declare Function TM1ValPoolGet Lib "tmlapi.dll" (ByVal hPool As Long, ByVal index As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. index is a long. This TM1 value capsule contains an integer that specifies the relative position (zero-based) of a value within the pool.
Result	Returns a value capsule. The value of the value capsule is the value handle at the position given by index in the value pool. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValPoolMemory

Retrieves the amount of memory in bytes currently used by a value pool.

Item	Description
Purpose	Retrieves the amount of memory in bytes currently used by a value pool.
Definition	<pre>Declare Function TM1ValPoolMemory Lib "tm1api.dll" (ByVal hPool As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate.
Result	This function returns an unsigned long containing the size of the value pool in kilobytes. The initial size of the value pool is 1 kilobyte.
Security	None.
Errors	None.
See Also	Other TM1ValPool functions.

TM1ValReal

To construct a value capsule containing a real value.

Item	Description
Purpose	To construct a value capsule containing a real value.

Item	Description
Definition	Declare Function TM1ValReal Lib "tmlapi.dll" (ByVal hPool As Long, ByVal InitReal As Double) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. InitReal is the value to be stored in the capsule.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ValRealGet

Retrieves the contents of a real value capsule.

Item	Description
Purpose	Retrieves the contents of a real value capsule.
Definition	Declare Function TM1ValRealGet Lib "tmlapi.dll" (ByVal hUser As Long, ByVal vReal As Long) As Long
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. vReal is a long. This IBM Cognos TM1 value capsule contains a real.
Result	The function returns the real contents of the value capsule. If there is an error, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ValRealSet

Update the contents of a real value capsule.

Item	Description
Purpose	Update the contents of a real value capsule.
Definition	Declare Sub TM1ValRealSet Lib "tmlapi.dll" (ByVal vReal As Long, ByVal Real As Double)
Parameters	vReal is a long. This is a value capsule whose contents is to be updated. Real is the value used to update the value capsule.
Result	This function does not return a value. The value of Real is stored in vReal.
Security	None.
Errors	None
See Also	Other TM1Val functions.

TM1ValString

Constructs a value capsule containing a string.

Item	Description
Purpose	Constructs a value capsule containing a string.
Definition	Declare Function TM1ValString Lib "tmlapi.dll" (ByVal hPool As Long, ByVal InitString As String, ByVal MaxSize As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. InitString is a string to be stored in the capsule. Maxsize is a long. This IBM Cognos TM1 value capsule contains a handle to an integer indicating the maximum length of a string that this value capsule can hold. A value of zero means that the maximum length should be equal to the length of the string being passed in.

Item	Description
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringEncrypt

Constructs a value capsule containing an encrypted string.

Item	Description
Purpose	Constructs a value capsule containing an encrypted string.
Definition	<pre>TM1IMPORT TM1V TM1API TM1ValStringEncrypt(TM1P hPool, CHAR * InitString, TM1_INDEX MaxSize);</pre>
Parameters	hPool is a valid pool handle obtained with TM1ValPoolCreate. InitString is a character array. This is password to be stored in the capsule. Maxsize is an integer indicating the maximum length of a string than can be held in this value capsule. A value of zero means that the maximum length should be equal to the length of the string being passed in.
Result	The function returns the handle to the value capsule created. If the value cannot be created, perhaps because of a lack of memory, the function returns zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringGet_VB

Retrieves a Microsoft Visual Basic string from a string value capsule.

Item	Description
Purpose	Retrieves a Microsoft Visual Basic string from a string value capsule.
Definition	Declare Sub TM1ValStringGet_VB(ByVal hUser as Long, ByVal sValue as Long, ByVal Retval as String, ByVal Max as Integer)
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. sValue is a long. This IBM Cognos TM1 value capsule contains a string value. Retval is a string where the result is returned. Max is the maximum length of string that can be accepted by Retval. Max must be numeric.
Result	The string Retval is returned padded with blanks. It must be declared with a fixed maximum length. The length should match the value of Max. For example: Dim Retval as String * 75 TM1ValStringGet_VB(hUser, sValue, Retval, 75)
Security	None.
Errors	None.

TM1ValStringMaxSize

Returns the maximum string size that can be stored in a string capsule.

Item	Description
Purpose	Returns the maximum string size that can be stored in a string capsule.
Definition	Declare Function TM1ValStringMaxSize Lib "tm1api.dll" (ByVal hUser As Long, ByVal vString As Long) As Long
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. vString is a long. This IBM Cognos TM1 value capsule contains a string value.

Item	Description
Result	The function returns the size of the longest string that can be saved in the string value capsule. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValStringSet

Update the contents of a string value capsule.

Item	Description
Purpose	Update the contents of a string value capsule.
Definition	Declare Sub TM1ValStringSet Lib "tmlapi.dll" (ByVal vString As Long, ByVal sString As String)
Parameters	vString is a long. This is the value capsule whose contents is to be updated. sString is a string that is used to update the capsule. The length of the new string value should not exceed the maximum length of the string specified when the capsule was originally created. If it does exceed this length, the new value is truncated accordingly.
Result	This function does not return a value. The value of String is inserted into the vString variable.
Security	None.
Errors	None.
See Also	Other TM1ValString functions.

TM1ValType

Retrieves the type of a value.

Item	Description
Purpose	Retrieves the type of a value.

Item	Description
Definition	Declare Function TM1ValType Lib "tm1api.dll" (ByVal hUser As Long, ByVal Value As Long) As Integer
Parameters	hUser is a long. This is a valid user handle obtained with Integrated Login. Value is a long. This is the value capsule whose type is to be retrieved.
Result	The function returns one of the following constants: TM1ValTypeReal(); TM1ValTypeString(); TM1ValTypeIndex(); TM1ValTypeBool(); TM1ValTypeObject(); TM1ValTypeError(); TM1ValTypeArray(); For example, if the value capsule returns TM1ValTypeReal(), the data in the value capsule is real. If there is an error, the function returns a zero.
Security	None.
Errors	None.
See Also	Other TM1Val functions.

TM1ViewArrayColumnsNof

Returns the number of columns in the view array including columns for dimensions and data.

Item	Description
Purpose	Returns the number of columns in the view array including columns for dimensions and data.
Definition	Declare Function TM1ViewArrayColumnsNof Lib "tm1api.dll" (ByVal hPool As Long, ByVal hView As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hView is a long. This IBM Cognos TM1 value capsule contains a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube property TM1CubeViews.

Item	Description
Result	Returns the number of columns in a view.
Security	None.
Errors	None.
See Also	TM1ViewArrayRowsNof

TM1ViewArrayConstruct

Constructs a two dimensional array of data that can be used to display the data of a view.

Item	Description
Purpose	Constructs a two dimensional array of data that can be used to display the data of a view. When you use TM1ViewArrayConstruct to access a registered view (via the hView parameter), IBM Cognos TM1 applies a server lock while the view is calculated.
Definition	<pre>Declare Function TM1ViewArrayConstruct Lib "tm1api.dll" (ByVal hPool As Long, ByVal hView As Long) As Long</pre>
Parameters	hPool is a long. This TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hView is a long. This TM1 value capsule contains a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
Result	Returns a long integer. The long contains a value capsule. Pass the value capsule to TM1ValBoolGet to retrieve the call result. If the result is a 1, the values in the view are available for retrieval. To extract values from a view, you would typically call functions in this order: <pre>TM1ViewArrayConstruct() TM1ViewArrayRowsNof() TM1ViewArrayColumnsNof() TM1ViewArrayValueGet()</pre>
Security	None.
Errors	None.

Item	Description
See Also	TM1ViewArrayValueGet TM1ViewArrayRowsNof TM1ViewArrayColumnsNof

TM1ViewArrayDestroy

Destroys view array constructed by TM1ViewArrayConstruct.

Item	Description
Purpose	Destroys view array constructed by TM1ViewArrayConstruct.
Definition	Declare Function TM1ViewArrayDestroy Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hView As Long) As Long
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hView is a long. This TM1 value capsule contains a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
Result	Destroys a view array created with the function TM1ViewArrayCreate.
Security	None.
Errors	None.
See Also	TM1ViewCreate

TM1ViewArrayRowsNof

Returns the number of rows in the view including rows for dimensions and data.

Item	Description
Purpose	Returns the number of rows in the view including rows for dimensions and data.
Definition	Declare Function TM1ViewArrayRowsNof Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hView As Long) As Long

Item	Description
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hView is a long. This TM1 value capsule contains a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube property TM1CubeViews.
Result	Returns the number of rows in a view array.
Security	None.
Errors	None.
See Also	TM1ViewArrayColumnsNof

TM1ViewArrayValueGet

Retrieves a single value from a view.

Item	Description
Purpose	Retrieves a single value from a view.
Definition	<pre>Declare Function TM1ViewArrayValueGet Lib "tm1api.dll" (ByVal hPool As Long, ByVal hView As Long, ByVal iColumn As Long, ByVal iRow As Long) As Long</pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hView is a long. This TM1 value capsule contains a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube property TM1CubeViews. iColumn is a long. This TM1 value capsule contains an integer. The integer is a 1-based number corresponding to the column of the value you want to retrieve. iRow is a long. This TM1 value capsule contains a long integer. The long integer is a 1-based number corresponding to the row of the value you want to retrieve.
Result	Returns a long. This TM1 value capsule contains a single cell value from a view. This value can be a number value, a string value, or a dimension element handle, or NULL. The dimension element could be either a column or row title. You must construct a view array by calling TM1ViewArrayConstruct before you can successfully call this function.

Item	Description
Security	None.
Errors	None.
See Also	TM1ViewArrayConstruct

TM1ViewCreate

Creates a view for a cube.

Item	Description
Purpose	Creates a view for a cube. A view is implemented as a sub-object of a cube.
Definition	<pre> Declare Function TM1ViewCreate Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hCube As Long, ByVal hTitleSubsetArray As Long, ByVal hColumnSubsetArray As Long, ByVal hRowSubsetArray As Long) As Long </pre>
Parameters	hPool is a long. This IBM Cognos TM1 value capsule contains a pool handle. The pool handle is returned by the function TM1ValPoolCreate. hCube is a long. This TM1 value capsule contains a valid cube handle obtained with TM1CubeCreate or from one of the TM1ListObject functions. hTitleSubsetArray is a long. This TM1 value capsule contains a handle to an array of subset handles. In Perspectives, this list of subsets appears under the static dimensions in a view. If there is more than one title dimension, the array will have more than one element.

Item	Description
Parameters (cont.)	This is a view of the 94sales cube. To create this view, you would create two subsets: A subset of the actvsbud dimension, one element of which is Variance, as displayed in the example. A subset of the region dimension, one element of which is World, as displayed in the example. Create an array that contains handles to these two subsets, then pass a handle to that array as the hTitleSubsetArray argument. hColumnSubsetArray is a long. This TM1 value capsule contains a handle to an array of subset handles. These subsets are displayed along the columns of the view. In the example, two subsets are passed: one subset of the month dimension, and one subset of the model dimension. hRowSubsetArray is a long. This TM1 value capsule contains a handle to an array of subset handles. These subsets are displayed in the rows of the view. In the example, a subset of the account1 dimension is passed.
Result	Creates a new view, and returns a handle to it. If all the elements of a dimension are to be used for the view do not use a subset handle. Use the handle of the subset's dimension instead.
Security	None.
Errors	None.
See Also	Other TM1View functions.

TM1ViewExtractCreate

Creates a sequential list of records from a view.

Item	Description
Purpose	Creates a sequential list of records from a view.
Definition	Declare Function TM1ViewExtractCreate Lib "tm1api.dll" (ByVal hPool As Long, ByVal hView As Long) As Long
Parameters	hPool is a pool handle obtained with TM1ValPoolCreate. hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views can be retrieved using the list properties of TM1CubeViews.
Result	Return a TM1V containing a handle to the view extract. The view extract object type is TypeOldQuery.

Item	Description
Security	None.
Errors	None.
See Also	TM1ViewExtractDestroy TM1ViewExtractGetNext

TM1ViewExtractDestroy

Destroys a view extract created by TM1ViewExtractCreate.

Item	Description
Purpose	Destroys a view extract created by TM1ViewExtractCreate.
Definition	Declare Function TM1ViewExtractDestroy Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hView As Long) As Long
Parameters	<i>hPool</i> is a pool handle obtained with TM1ValPoolCreate. <i>hView</i> is a handle to an extract. A handle to an extract is returned by the function TM1ViewExtractCreate.
Result	Returns a TM1V containing a TM1_BOOL. If the Boolean is TRUE (1), the function executed successfully.
Security	None.
Errors	None.
See Also	TM1ViewExtractCreate TM1ViewExtractGetNext

TM1ViewExtractGetNext

Return the result of a view extract.

Item	Description
Purpose	Return the result of a view extract.
Definition	Declare Function TM1ViewExtractGetNext Lib "tmlapi.dll" (ByVal hPool As Long, ByVal hView As Long) As Long
Parameters	<i>hPool</i> is a pool handle obtained with TM1ValPoolCreate. <i>hView</i> is a handle to an extract. A handle to an extract is returned by the function TM1ViewExtractCreate.

Item	Description
Result	Returns a TM1V. Element positions are returned as indexes, element. Names are returned as strings, and values are returned as reals or strings.
Security	None.
Errors	None.
See Also	TM1ViewExtractCreate TM1ViewExtractDestroy

Chapter 6. IBM Cognos TM1 properties

Properties are values associated with objects. Some properties are defined for all objects. Others are specific to a type of object.

The name of a property describes the type of object that a particular property applies to. The following list shows a series of prefixes that are used in property names, and the corresponding object type:

- TM1Client properties apply to users attached to an IBM Cognos TM1 server.
- TM1Cube properties apply to cubes.
- TM1Dimension properties apply to dimensions.
- TM1Element properties apply to elements.
- TM1Object properties apply to all TM1 objects.
- TM1Rule properties apply to rules.
- TM1Subset properties apply to subsets.
- TM1View properties apply to views.

All the TM1 properties are presented in alphabetical order here.

TM1AttributeType property

Item	Description
Purpose	Contains a constant indicating the type of attribute, as follows: TM1TypeAttributeAlias(void); TM1TypeAttributeString(void); TM1TypeAttributeNumeric(void); TM1TypeAttributeBoolean(void);
Type	TM1_INDEX
Updateable	No
Comments	The attribute type is determined when the attribute is created with TM1ObjectAttributeInsert. It cannot be changed.

TM1BlobSize property

Item	Description
Purpose	A number. This is the size of the BLOB in bytes.

Item	Description
Type	TM1_INDEX
Updateable	The server updates this property as the BLOB changes size.

TM1ClientPassword property

Item	Description
Purpose	Contains a Client's current password.
Type	TM1_STRING
Updateable	Yes -- by the Client and the Administrator. The default value is blank.

TM1ClientStatus property

Item	Description
Purpose	Contains a client's status.
Type	TM1_B00L
Updateable	Yes -- by the Client and the Administrator. The default value is blank.
Comments	A return of 1 means the client and server are connected. A return of 0 means the client and server are not connected.

TM1ConnectionChoresUsing property

Item	Description
Purpose	This list object contains a set of chore objects. You can set up a chore to execute a synchronization on an existing connection object. For every chore that schedules a connection to be synchronized, an object is added to this list.
Type	A collection of chore objects.

Item	Description
Updateable	No. You cannot update this list directly. However, you can create a chore with TM1ChoreCreateEmpty.

TM1ConnectionLastSyncTime property

Item	Description
Purpose	The time (GMT) at which data was last synchronized on the planet server on which the replication connection was established. The property is stored in yyyyymmddhhss format.
Type	A TM1_String containing the time and date stamp.
Updateable	No.

TM1ConnectionLastSyncTimeStar property

Item	Description
Purpose	The time (GMT) at which data was last synchronized on the star server from which the replication connection was established. The property is stored in yyyyymmddhhss format.
Type	A TM1_String containing the time and date stamp.
Updateable	No.

TM1ConnectionUsername property

Item	Description
Purpose	Contains the name of the user passed in the sUsername argument when you called TM1ConnectionCreate.
Type	TM1V containing a string.
Updateable	No.

TM1ConnectionSyncErrorCount property

Item	Description
Purpose	This property contains the number of errors generated during the last synchronization on this connection.
Type	TM1_Index
Updateable	No.

TM1ConnectionSyncPlanetToStar property

Item	Description
Purpose	If set to TRUE, data is moved from the planet server to the star server during a synchronization.
Type	TM1_Boolean
Updateable	Yes. Set this the TRUE or FALSE before calling TM1ConnectionSynchronize. If you want data to be synchronized in two directions, set TM1ConnectionSyncStarToPlanet in addition to this property.

TM1ConnectionSyncStarToPlanet property

Item	Description
Purpose	If set to TRUE, data is moved from the star server to the planet server during a synchronization.
Type	TM1_Boolean
Updateable	Yes. Set this the TRUE or FALSE before calling TM1ConnectionSynchronize. If you want data to be synchronized in two directions, set TM1ConnectionSyncPlanetToStar in addition to this property.

TM1CubeCellValueUndefined property

Item	Description
Purpose	Returns an undefined value handle to use in setting cells and views.
Type	TM1_OBJECT containing a value handle.
Updateable	No

TM1CubeDimensions property

Item	Description
Purpose	Contains a list of the dimensions in a cube.
Type	List of dimension handles .
Updateable	No

TM1CubeLogChanges property

Item	Description
Purpose	Contains a Boolean. This value is 1 if the changes to a cube are to be stored in the change log file.
Type	TM1_BOOL
Updateable	Yes. The default value is 1, which means all cube changes are logged by default. If you change this property to 0, no cube changes are logged.
Comments	This flag may be updated at any time to stop or resume logging changes to a cube.

TM1CubeMeasuresDimension property

Item	Description
Purpose	Contains the measures dimension for a cube.
Type	TM1_OBJECT containing a dimension handle.
Updateable	Yes

Item	Description
Comments	Set it and retrieve a TM1V.

TM1CubePerspectivesMaxMemory property

Item	Description
Purpose	Contains the maximum number of bytes that can be used to store views. If this value is set to 0, no views will be stored permanently for this cube.
Type	TM1_INDEX
Updateable	Yes

TM1CubePerspectivesMinTime property

Item	Description
Purpose	Contains the number of seconds required to calculate a view, below which the view will not be stored. The recommended value is 5. The default for this property is 5.
Type	TM1_INDEX
Updateable	Yes

TM1CubeReplicationSyncRule property

Item	Description
Purpose	If set to TRUE, this cube's rule will be updated when the cube is synchronized.
Type	TM1_Boolean
Updateable	Yes. Set this to TRUE or FALSE before calling TM1ObjectReplicate for the cube.

TM1CubeReplicationSyncViews property

Item	Description
Purpose	If set to TRUE, this cube's views will be updated when the cube is synchronized.
Type	TM1_Boolean
Updateable	Yes. Set this to TRUE or FALSE before calling TM1ObjectReplicate for the cube.

TM1CubeRule property

Item	Description
Purpose	Contains the handle to the rule that applies to a cube.
Type	TM1_OBJECT
Updateable	Yes. Use the TM1Rule functions to update this property.

TM1CubeTimeDimension property

Item	Description
Purpose	Contains a handle to the time dimension for a cube.
Type	TM1_OBJECT
Updateable	Yes
Error	TM1ErrorCubeNoTimeDimension

TM1CubeViews property

Item	Description
Purpose	Contains a list of the public or private views in a cube.
Type	List of view handles .

Item	Description
Updateable	Views are added to a cube when you call TM1ObjectRegister or TM1ObjectPrivateRegister. They are deleted from the cube when you call TM1ObjectDelete for the given view. You should not directly manipulate this property.

TM1DimensionCubesUsing property

Item	Description
Purpose	Contains a list of cubes that are using the current dimension.
Type	List of object handles
Updateable	No
Comment	You can not delete a dimension if a cube is using it. This property contains a list of every cube on the IBM Cognos TM1 server that uses this dimension.

TM1DimensionElements property

Item	Description
Purpose	Contains a list of the elements in a dimension
Type	List of element handles .
Updateable	No
Comment	This list is 1-based. The first element in the list is at index 1.

TM1DimensionNofLevels property

Item	Description
Purpose	Contains a numeric value stating the number of consolidation levels in a dimension.
Type	TM1_INDEX

Item	Description
Updateable	No

TM1DimensionReplicationSyncSubsets property

Item	Description
Purpose	If set to TRUE, this dimension's subsets will be updated when the cube is synchronized.
Type	TM1_Boolean
Updateable	Yes. Set this to TRUE or FALSE before calling TM1ObjectReplicate for the dimension.

TM1DimensionSubsets property

Item	Description
Purpose	Contains a list of public and private subsets in a dimension.
Type	List
Updateable	No. Subsets are added to a dimension when you call TM1ObjectRegister or TM1ObjectPrivateRegister. They are deleted from the dimension when you call TM1ObjectDelete. You should not directly manipulate this property.

TM1DimensionTopElement property

Item	Description
Purpose	The first element added to a dimension when it is created is the top element. This is the first element displayed in the dimension by the Perspectives dimension editor. Deprecated. This function will be removed from the product in a future release of IBM Cognos TM1. Rather than use this property as an argument to TM1ObjectPropertyGet, We recommend calling TM1ObjectListHandleByIndexGet and passing the following arguments: • A pool handle • A dimension handle • The property TM1DimensionElements() • An index of 1
Type	TM1_OBJECT
Updateable	No

TM1DimensionWidth property

Item	Description
Purpose	Contains a numeric value stating the length of the longest element in a dimension.
Type	TM1_INDEX
Updateable	No

TM1ElementComponents property

Item	Description
Purpose	Contains a list of the components of a consolidated element in a dimension.
Type	List of element handles
Updateable	This is updateable only by adding or deleting elements of a dimension consolidation - not by direct manipulation.

TM1ElementIndex property

Item	Description
Purpose	Contains the position of an element within a dimension.
Type	TM1_INDEX
Updateable	No

TM1ElementLevel property

Item	Description
Purpose	A number indicating the consolidation level of an element within a dimension. Simple elements have level 0. The level of a calculated element is the maximum level of its components plus one.
Type	TM1_INDEX
Updateable	No

TM1ElementParents property

Item	Description
Purpose	When used with the TM1ObjectListCountGet function, a request for this property returns the number of parents the element has. When used with TM1ObjectListHandleByIndexGet, a request for this property returns the parent handle corresponding to the index specified.
Type	List of parent handles . If an element appears more than once in a hierarchical dimension, you will get a list with more than one handle.
Updateable	No

TM1ElementType property

Item	Description
Purpose	Contains a numeric code corresponding to the type of an element.
Type	TM1_INDEX which can be compared against one of the following constants: TM1TypeElementSimple() TM1TypeElementConsolidated() TM1TypeElementString()
Updateable	No

TM1ObjectAttributes property

Item	Description
Purpose	Contains a list of an object's attributes.
Type	List of attribute handles
Updateable	No

TM1ObjectChangedSinceLoaded property

Item	Description
Purpose	Contains a Boolean flag stating whether or not the object has been changed since it was loaded from disk.
Type	TM1_BOOL
Updateable	No. The server updates this property when the user makes a change to the object.
Comment	You can call this for any object on the server except the server itself. You can call it on a server handle, cube handle, or dimension handle. Any other handle will result in TM1ErrorObjectPropertyNotDefined.

TM1ObjectLastTimeUpdated property

Item	Description
Purpose	Contains a string indicating the last time the data for this object was updated.
Type	A TM1V containing a string.
Updateable	No. The server updates this property when the user makes a change to the object.
Comment	You can call this for any object on the server except the server itself. Call it on a cube handle, rule handle, or dimension handle. Any other handle will result in TM1ErrorObjectPropertyNotDefined.

TM1ObjectMemoryUsed property

Item	Description
Purpose	Contains the number of bytes of memory used to store an object. Call it on the following objects: dimension, server, rule, subset, cube or view.
Type	TM1_INDEX
Updateable	No
Comment	Returns TM1_INDEX indicating the number of bytes used.

TM1ObjectName property

Item	Description
Purpose	Contains a string with the name of an object.
Type	String
Updateable	No

TM1ObjectNull property

Item	Description
Purpose	Sets a new handle to TM1ObjectNull() to initialize it.
Type	TM1_OBJECT
Updateable	No.

TM1ObjectParent property

Item	Description
Purpose	Contains the handle of the parent of an object.
Type	TM1_OBJECT
Updateable	No

TM1ObjectRegistration property

Item	Description
Purpose	Contains a TM1V containing an index. The index indicates the registration status of the object.
Type	TM1_INDEX
Updateable	No
Comment	Returns the registration status of an object. This means if the object is registered, private, public, or unregistered. TM1V contains one of these three values: TM1ObjectPublic() TM1ObjectPrivate() TM1ObjectUnregistered()

TM1ObjectReplicationConnection property

Item	Description
Purpose	Takes the connection object handle on which this object is replicated.
Type	A connection object handle , as returned from TM1ConnectionCreate or the IBM Cognos TM1 server list property TM1ServerConnections.
Updateable	Yes. Set this property before you call TM1ObjectReplicate.

TM1ObjectReplicationSourceObjectName property

Item	Description
Purpose	Takes an IBM Cognos TM1 value capsule containing a string. The string is the name of the source object from which this object is replicated.
Type	A TM1V containing a string.
Updateable	Yes. Set this property before you call TM1ObjectReplicate.

TM1ObjectReplicationStatus property

Item	Description
Purpose	Contains the status of a replicated object. Possible values include copied, indicating that the object has been copied from the star server to the planet server, and not copied. Before you call TM1ObjectReplicate for a cube object, you must set this property to not copied for the cube.
Type	A TM1V containing a string.
Updateable	Yes. Set this property before you call TM1ObjectReplicate for the first time on a cube object.

TM1ObjectSecurityOwner property

Item	Description
Purpose	Contains the handle of the name of the Client who has locked or reserved an object.
Type	TM1_STRING
Updateable	No
Comments	If the object is not locked the TM1ObjectPropertyGet function returns a Boolean 0.

TM1ObjectSecurityStatus property

Item	Description
Purpose	Contains the security status of the current object.
Type	TM1_STRING A TM1V containing a string.
Updateable	No
Comments	Returns a string that says LOCKED, AVAILABLE, or RESERVED.

TM1ObjectType property

Item	Description
Purpose	Integer indicating type of object.
Type	TM1_INDEX
Updateable	No

Item	Description
Comments	The IBM Cognos TM1 API defines functions that return integers corresponding to a particular object type. See the tmlapi.h file for more information. TM1ObjectType contains the following constants: TM1TypeAttribute() TM1TypeAttributeAlias() TM1TypeAttributeBoolean() TM1TypeAttributeNumeric() TM1TypeAttributeString() TM1TypeBlob() TM1TypeClient() TM1TypeCube() TM1TypeDimension() TM1TypeElement() TM1TypeElementConsolidated() TM1TypeElementSimple() TM1TypeElementString() TM1TypeGroup() TM1TypeRule() TM1TypeRuleCalculation() TM1TypeRuleDrill() TM1TypeServer() TM1TypeSubset() TM1TypeView()

TM1RuleErrorLine property

Item	Description
Purpose	Contains the first line of a rule where a compilation error has occurred.
Type	TM1_INDEX
Updateable	No. This is updated when you call the function TM1RuleCheck.

TM1RuleErrorString property

Item	Description
Purpose	Contains the text of the last compilation error in a rule.

Item	Description
Type	String
Updateable	No. This is updated when you call the function TM1RuleCheck.

TM1RuleNofLines property

Item	Description
Purpose	Contains the number of lines in a rule.
Type	TM1_INDEX
Updateable	No

TM1ServerBlobs property

Item	Description
Purpose	Contains a list of the Blobs in a server.
Type	List of Blob handles
Updateable	No

TM1ServerBuildNumber property

Item	Description
Purpose	Contains the build number of the IBM Cognos TM1 server. This information may be required for troubleshooting purposes. It is displayed in the TM1 Server monitor when you run it as a desktop application:
Type	This property is a TM1V containing a string.
Updateable	No

TM1ServerChores property

Item	Description
Purpose	Contains a list of chores on an IBM Cognos TM1 server.
Type	List of chore handles
Updateable	No

TM1ServerClients property

Item	Description
Purpose	Contains a list of the clients in a server
Type	List of client handles
Updateable	No

TM1ServerConnections property

Item	Description
Purpose	Contains the server's list of connection handles.
Type	List of connection object handles . Use TM1ObjectListHandlebyNameGet or TM1ObjectListHandlebyIndexGet to retrieve a connection handle from the list.
Updateable	No. This cannot be directly changed. When you add connection by calling TM1ConnectionAdd, a connection object handle is added to this list. When you delete a connection with TM1ConnectionDelete, a connection object is deleted from this list.

TM1ServerCubes property

Item	Description
Purpose	Contains a list of the cubes in a server
Type	List of cube handles

Item	Description
Updateable	No

TM1ServerDimensions property

Item	Description
Purpose	Contains a list of the dimensions in a server.
Type	List of dimension handles
Updateable	No

TM1ServerDirectories property

Item	Description
Purpose	Contains a string of directories, separated by semicolons, where the Server stores and retrieves permanent objects.
Type	String
Updateable	No

TM1ServerGroups property

Item	Description
Purpose	Contains a list of the groups in a server.
Type	List of group handles
Updateable	No

TM1ServerLogDirectory property

Item	Description
Purpose	Contains directory and path for server log files.
Type	TM1_STRING

Item	Description
Updateable	No

TM1ServerNetworkAddress property

Item	Description
Purpose	Contains the IP address of the server.
Type	TM1_STRING
Updateable	No

TM1ServerProcesses property

Item	Description
Purpose	Contains a list of processes on an IBM Cognos TM1 server.
Type	List of process handles
Updateable	No

TM1SQLTableColumnNames property

Item	Description
Purpose	Contains an array containing the names of the columns in a TM1SqlTableObject.
Type	TM1Val array
Updateable	No.

TM1SQLTableColumnType property

Item	Description
Purpose	Identifies the type of data contained in each column of a TM1SqlTableObject.

Item	Description
Type	An array of TM1Val. The IBM Cognos TM1 value capsules contain one of the following values: TM1TypeSQLNumericColumn TM1TypeSQLStringColumn TM1TypeSQLNotSupported
Updateable	No.

TM1SQLTableNumberOfColumns property

Item	Description
Purpose	Contains the number of columns in a TM1SqlTableObject.
Type	TM1_Index
Updateable	No.

TM1SQLTableNumberOfRows property

Item	Description
Purpose	Contains the number of rows in a TM1SqlTableObject.
Type	TM1_Index
Updateable	No.

TM1SQLTableRowsetSize property

Item	Description
Purpose	Sets the number of rows to be returned by the function TM1SQLTableGetNextRows.
Type	TM1_Index
Updateable	Yes. You should set this property before calling TM1SQLTableGetNextRows. If you do not set this property, TM1SQLTableGetNextRows returns 50 rows.

TM1SubsetAlias property

Item	Description
Purpose	If this property is NULL, no element in the subset contains an alias. If an element in the subset contains an alias, this property contains the name of the alias attribute. You must examine the elements in the dimension to extract the alias names.
Type	String
Updateable	No

TM1SubsetElements property

Item	Description
Purpose	An array of subset handles. These subsets are displayed in the columns of the view. You can access this property by calling <code>TM1ObjectListHandleByIndexGet</code> . You cannot access this property by calling <code>TM1ObjectListHandleByNameGet</code> .
Type	An array of element handles
Updateable	No

TM1SubsetExpression property

Item	Description
Purpose	Contains the MDX expression used to create the subset. This property is populated only for subsets created through <code>TM1SubsetCreateByExpression</code> .
Type	String
Updateable	No

TM1ViewColumnSubsets property

Item	Description
Purpose	An array of subset handles. These subsets are displayed in the columns of the view.
Type	An array of subset objects
Updateable	No

TM1ViewExtractComparison property

Item	Description
Purpose	Sets the comparison operators as follows: TM1ViewExtractComparisonNone(void); TM1ViewExtractComparisonEQ_A(void); TM1ViewExtractComparisonGE_A(void); TM1ViewExtractComparisonLE_A(void); TM1ViewExtractComparisonGT_A(void); TM1ViewExtractComparisonLT_A(void); TM1ViewExtractComparisonNE_A(void); TM1ViewExtractComparisonGE_A_LE_B (void); TM1ViewExtractComparisonGT_A_LT_B (void);
Type	TM1_INDEX
Updateable	Yes

TM1ViewExtractRealLimitA property

Item	Description
Purpose	Sets the lower end of the real number limit for a view extract.
Type	TM1_REAL
Updateable	Yes

TM1ViewExtractRealLimitB property

Item	Description
Purpose	Sets the upper end of the real number limit for a view extract.
Type	TM1_REAL
Updateable	Yes

TM1ViewExtractSkipConsolidatedValues property

Item	Description
Purpose	Sets the view properties to skip consolidated (non - rules) values.
Type	TM1_B00L
Updateable	Yes

TM1ViewExtractSkipRuleValues property

Item	Description
Purpose	Sets the view properties to skip rule-calculated values.
Type	TM1_B00L
Updateable	Yes

TM1ViewExtractSkipZeroes property

Item	Description
Purpose	Skip zero values in a view extract.
Type	TM1_B00L
Updateable	Yes

TM1ViewExtractStringLimitA property

Item	Description
Purpose	Sets the lower end of the text limit for a view extract.
Type	TM1_STRING
Updateable	Yes

TM1ViewExtractStringLimitB property

Item	Description
Purpose	Sets the upper end of the text limit for a view extract.
Type	TM1_STRING
Updateable	Yes

TM1ViewFormat property

Item	Description
Purpose	Contains a string that indicates the applied format of the cube view. If the string is NULL, this view has no format. The string has the following format: <pre><format_code>:<format_string> [<form_feed><dialog_helper_str>]</pre> Where the <i>format_code</i> is a single character - either b or c. The <i>format_string</i> is a digit template. This template varies depending on the format chosen for the view. Examples of each possible format are shown below. The <i>dialog_helper_str</i> is something like F 2 Y. These strings are for use by the IBM Cognos TM1 Server Explorer. Deprecated: This property is obsolete and will be removed from the TM1 product in a future release. Use the property TM1ViewFormatString in place of this property in all new TM1 applications.

Item	Description
Examples	General: b:0.##### G 0
	Fixed: b:0.00;(0.00) F 2 Y
	Comma: b:##,##0.00;(#,##0.00
	Currency: b:\\$#,###.00;(\\$#,##
	Date: b:dddd, mmmm dd, yy
	Time: b:h:nn:ss AMPM T 0
	Percent: b:##.00% P 2
	Scientific: b:##.00E+## S 2
	Custom: c:####.##
Type	TM1_B00L
Updateable	No

TM1ViewFormatString property

Item	Description
Purpose	Contains a string that indicates the applied format of the cube view. The string is a digit template. This template varies depending on the format chosen for the view. Examples of each possible format are shown below. If the string is NULL, this view has no format.

Item	Description
Examples	General: 0.#####
	Fixed: 0.00;(0.00)
	Comma: #,##0.00;(#,##0.00
	Currency: \\$#,###.00;(\\$#,##
	Date: dddd, mmmm dd, yyy
	Time: h:nn:ss AMPM
	Percent: #.00%
	Scientific: #.00E+##
	Custom: ####.##
Type	TM1_B00L
Updateable	No

TM1ViewPreConstruct property

Item	Description
Purpose	A Boolean. If TRUE, values for the view are calculated when the server initializes. Otherwise, the values for the view must be calculated when the view is displayed.
Type	TM1_B00L
Updateable	No

TM1ViewRowSubsets property

Item	Description
Purpose	Contains an array of subset handles that are displayed in the rows of the view.
Type	An array of row handles
Updateable	Yes

TM1ViewShowAutomatically property

Item	Description
Purpose	A Boolean. If TRUE, the view is automatically re-displayed in Perspectives when the view is re-configured.
Type	TM1_B00L
Updateable	Yes

TM1ViewSuppressZeroes property

Item	Description
Purpose	A Boolean. If TRUE, rows and columns containing only zeroes are not displayed in the view.
Type	TM1_B00L
Updateable	Yes

TM1ViewTitleElements property

Item	Description
Purpose	Contains an array of element handles that are displayed in the view. The number of elements in this array is always the same as the number of elements in the TM1ViewTitleSubsets array. The items in the two arrays also correspond - The first element in this array belongs to the first subset element in the TM1ViewTitleSubsets array; the second element in this array belongs to the second subset in the TM1ViewTitleSubsets array, and so on.
Type	An array of element handles .
Updateable	Yes

TM1ViewTitleSubsets property

Item	Description
Purpose	An array of subset handles. These elements are displayed in the view.
Type	List of subset handles
Updateable	No

Chapter 7. Performing data spreading with the IBM Cognos TM1 API

One of the most powerful and complex features of IBM Cognos TM1 is data spreading.

Data spreading allows you to write large amounts of data into your TM1 database very quickly. This feature is commonly used in planning and budgeting applications. For example, when you are projecting expenses based on last year's actual results, you can use data spreading to move last year's results, plus a small percentage, into this year's projected budget. This gives you a baseline for projecting expenses for the next fiscal period.

Performing data spreading through the API involves only four functions:

- TM1CubeCellSpread
- TM1CubeCellSpreadViewArray
- TM1CubeCellSpreadStatusGet
- TM1ClientHasHolds

These functions allow you to perform eleven types of data spreading. The spreading functions are described in detail in the *TM1 Users Guide*. This section describes how to call each of these spread types through the TM1 API functions:

- Spreading Overview
- Spreading Internals
- The Spreading Function Arguments
- Spreading Control Codes

Spreading overview

You must be aware of certain things when working with the IBM Cognos TM1 API.

In order to successfully execute data spreading from within the IBM Cognos TM1 API, you must know all of the following information:

- The type of spreading that you want to perform. There are eleven different kinds of spreading in TM1. The arguments that you supply to TM1CubeCellSpread and TM1 CubeCellSpreadViewArray vary depending on the type of spreading you choose.
- Whether or not you are spreading to a view. If you have retrieved a view handle from a TM1 cube, and you want to spread data to a portion of that view, you must call TM1CubeCellSpreadViewArray. If you want to spread data to a portion of a cube, but you do not have a view handle, you can use TM1CubeCellSpread.
- The syntax of the spreading command you want to execute. Spreading commands are strings that you pass to the functions TM1CubeCellSpread (through the sSpreadData argument) and TM1CubeCellSpreadViewArray (through the sControl argument). The spreading command contains the value to be spread, information on what cells are affected by the spread, and what type of spreading you are using.

Spreading internals

This section discusses how spreading is resolved by the IBM Cognos TM1 server.

The rules for executing a spreading function across a cell range vary depending on the range of cells to which you are spreading data. There are three separate cases discussed here:

- Spreading to a single leaf cell
- Spreading to a single consolidated cell
- Spreading to a Range of cells

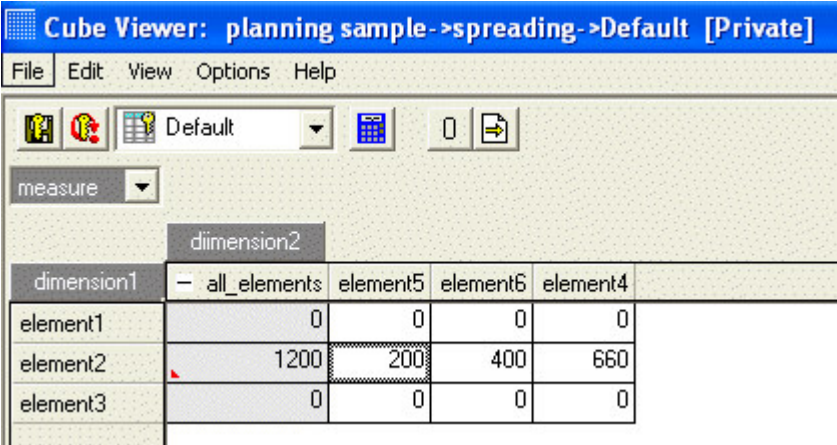
Each of these cases is described in the sections that follow.

Spreading to a single leaf cell

Spreading to a single leaf cell updates the cell unless it is not updatable because of security limitations or some other constraint, or if a hold is applied to it.

A leaf cell on hold is never updated by a spread function. It may be updated by manually changing the cell through the user interface, or calling `TM1CubeCellValueSet`.

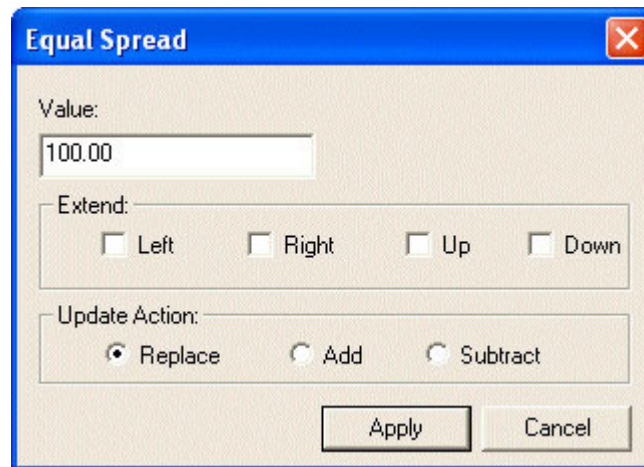
If one of the ancestors of the leaf cell is on consolidation hold, other leaf components of the ancestor are adjusted to keep the consolidated value constant. For example, consider the following figure.



The screenshot shows the 'Cube Viewer: planning sample->spreading->Default [Private]' window. It features a menu bar (File, Edit, View, Options, Help) and a toolbar with icons for selection, calculation, and navigation. Below the toolbar is a 'measure' dropdown set to 'Default'. The main area displays a table with two dimensions: 'dimension1' and 'dimension2'. 'dimension1' has elements 'element1', 'element2', and 'element3'. 'dimension2' has elements 'all_elements', 'element5', 'element6', and 'element4'. The table contains numerical values, with 'element2' under 'all_elements' having a value of 1200 and a red arrow icon next to it. The cell at the intersection of 'element2' and 'element5' is highlighted with a dotted border and contains the value 200.

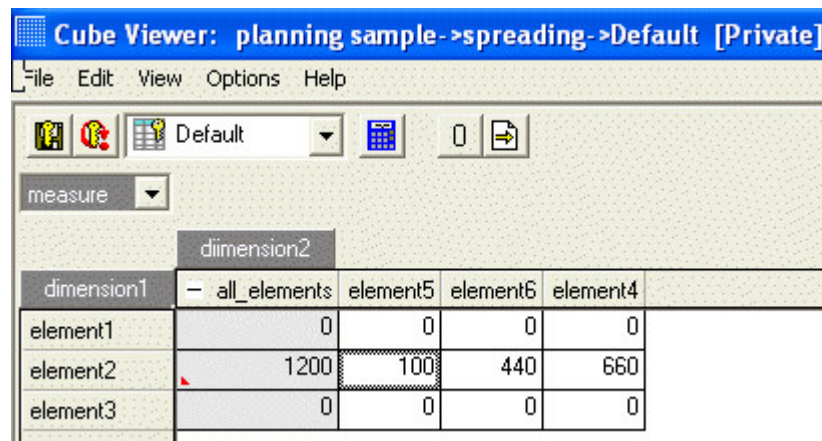
	dimension2			
dimension1	all_elements	element5	element6	element4
element1	0	0	0	0
element2	1200	200	400	660
element3	0	0	0	0

In this example, the intersection of element2 and All Elements has a consolidation hold applied to it. Suppose you execute the following spread function from the intersection of element5 and element2:



The 'Equal Spread' dialog box has a title bar with a close button. It contains a 'Value' field with '100.00'. Below it is an 'Extend' section with four checkboxes: 'Left', 'Right', 'Up', and 'Down', all of which are unchecked. Below that is an 'Update Action' section with three radio buttons: 'Replace' (which is selected), 'Add', and 'Subtract'. At the bottom are 'Apply' and 'Cancel' buttons.

In this case, the consolidated hold forces updates to the other cells in the consolidation, as shown in the following figure:



The screenshot shows the 'Cube Viewer' window for 'planning sample->spreading->Default [Private]'. It features a menu bar (File, Edit, View, Options, Help) and a toolbar with icons for various functions. Below the toolbar is a 'measure' dropdown set to 'Default'. The main area displays a pivot table with 'dimension1' as the row header and 'dimension2' as the column header. The table data is as follows:

dimension1	all_elements	element5	element6	element4
element1	0	0	0	0
element2	1200	100	440	660
element3	0	0	0	0

Spreading to a consolidated cell

With the exception of functions that act on leaves, such as Repeat Leaves or Equal Spread Leaves, spreading to a consolidated cell means proportionally spreading to all updateable leaves that have no holds applied.

The following steps illustrate the algorithm used by IBM Cognos TM1.

Procedure

1. Get the value of the consolidated cell before spreading.
2. Calculate the sum of all the consolidation's leaf cells that are updateable and do not have a hold applied.
3. Calculate the non-updateable sum by subtracting <2> from <1>.
4. Subtract <3> from the spreading value.
5. Spread <4> to the leaf cells that are updateable and do not have a hold applied. Spread to these cells in proportion to their values before the spreading.

Spreading to a range

When you spread data to a range, the cells in the range are filtered before the spread occurs.

Filter 1 - If the range to which the data will be spread contains cells of mixed levels of consolidation, all cells are dropped except the cells at the lowest level.

Filter 2a - If the lowest level in the range are leaf cells, cells that are not updateable because of security or other constraints are dropped. Similarly, cells that have holds applied to them are dropped.

Filter 2b - If the lowest level remaining in the range are consolidated cells, all cells that have a consolidated hold are dropped.

Spreading is performed on the remaining cells in the range.

After spreading is complete

Internally, IBM Cognos TM1 performs a series of checks to determine if a spreading function was performed properly.

Those checks are:

- The values after the spreading of all consolidated cells that participated in the spreading are compared to their desired values.
- The values before and after the spreading are compared for all cells with consolidate holds. If any of the comparisons failed to be within the predefined relative accuracy, the spreading is deemed failed and all changes are rolled back. The default for the predefined relative accuracy is 1e-8, but this value is configurable for the server through the TM1s.cfg parameter SpreadingPrecision.

The spreading function arguments

This section discusses the arguments for the functions TM1CubeCellSpread and TM1CubeCellSpreadViewArray.

```
TM1IMPORT TM1V TM1API TM1CubeCellSpreadViewArray( TM1P
hPool, TM1V hView, TM1V aCellRange, TM1V aCellRef, TM1V sControl);
TM1IMPORT TM1V TM1API TM1CubeCellSpread( TM1P hPool, TM1V hServer,
TM1V vArrayOfCells, TM1V vCellReference, TM1V sSpreadData);
```

The values required for these arguments vary according to the type of spreading you choose. The sections that follow describe each type of spreading, and how the arguments to these functions must be set to successfully spread data.

Proportional spread, equal spread and repeat

The proportional spread method distributes a specified value among cells proportional to existing cell values. The equal spread method distributes a specified value equally across cells in a view. The repeat method repeats a specified value across cells in a view.

All three of these spread functions use the same arguments. To call TM1CubeCellSpreadViewArray, and execute a proportional spread function, an equal spread function, or a repeat function, set the arguments as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.

Argument	Description
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect.
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
aCellRange	This argument is a TM1V containing an array of row and column pairs. There is either one pair of integers, or two pairs of integers. If the array contains one pair of integers, the intersection of the row and column pair is the starting cell for the spread. For example, suppose the array was constructed as follows: <pre>aCellRange = {Row, Column}Row = TM1ValIndex(hPool, 1) ;Column = TM1ValIndex(hpool, 1);</pre> This indicates that the starting point for the proportional spread is the top left-hand corner of the view. This location can be either a consolidated cell or a leaf cell.
	If the array contains two pairs of integers, the two pairs define a range of values. For example, the aCellRange array for a proportional spread could be constructed as follows: <pre>aCellRange = {Row1, Column1, Row2, Column2}; Row = TM1ValIndex(hPool, 1); Column= TM1ValIndex(hpool, 3); Row = TM1ValIndex(hPool, 3); Column = TM1ValIndex(hpool, 4);</pre>
aCellRef	Set this value to TM1ArrayNull(). This argument is not used for proportional spread, equal spread or repeat.
sControl	Refer to "Spreading control codes" on page 410 for complete information. Directional information is not required when aCellRange defines a range of cells. It is optional if aCellRange defines a single cell.

To call TM1CubeCellSpread, and execute a proportional spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	<i>hPool</i> is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	<i>hServer</i> is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
vArrayOfCells	<i>vArrayOfCells</i> is a TM1V containing an array cell references. This array of cell references takes the form: <pre> VArrayOfCells = {Array1, Array2, Array3..., Arrayn}; Array1 = {CubeHandle1, ElementHandle, ElementHandle...,ElementHandle}; Array2 = {CubeHandle2, ElementHandle, ElementHandle...,ElementHandle}; </pre> The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property <code>TM1DimensionElements</code>.
vCellReference	Set this value to <code>TM1ArrayNull()</code> . This argument is not used for proportional spread, equal spread or repeat.
sSpreadData	Refer to “Spreading control codes” on page 410 for complete information.

Clear

The Clear method clears values from cells in a view.

To call `TM1CubeCellSpreadViewArray`, and execute a clear function, the arguments must be set as described in the following table.

Argument	Description
hPool	<i>hPool</i> is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	<i>hServer</i> is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .

Argument	Description
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
aCellRange	This argument is a TM1V containing an array of row and column pairs. There is either one pair of integers, or two pairs of integers. If the array contains one pair of integers, the intersection of the row and column pair is the starting cell for the spread. For example, suppose the array was constructed as follows: <pre>aCellRange = {Row, Column}Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool, 1);</pre> This indicates that the starting point for the spread is the top left-hand corner of the view. This location can be either a consolidated cell or a leaf cell. If aCellRange defines a single consolidated cell, the entire consolidation will be cleared when the function executes successfully.
	If the array contains two pairs of integers, the two pairs define a range of values. For example, the aCellRange array for a proportional spread could be constructed as follows: <pre>aCellRange = {Row1, Column1, Row2, Column2}; Row = TM1ValIndex(hPool, 1);Column = TM1ValIndex(hpool, 3); Row = TM1ValIndex(hPool, 3); Column = TM1ValIndex(hpool, 4);</pre>
aCellRef	Set this value to TM1ArrayNull(). This argument is not used for a clear spread function.
sControl	The spread code for clear spread is C. Refer to "Spreading control codes" on page 410 for complete information. Directional information is not required when aCellRange defines a range of cells. It is optional if aCellRange defines a single cell.

To call `TM1CubeCellSpread`, and execute a clear spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
vArrayOfCells	vArrayOfCells is a TM1V containing an array cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1, Array2, Array3..., Arrayn}; Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle}; Array2 = {CubeHandle2, ElementHandle, ElementHandle..., ElementHandle};</pre> The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the element handles within the specified cube. Element handles are extracted from the TM1 dimension list property <code>TM1DimensionElements</code>. If a cell referenced by vArrayOfCells is consolidated, the entire consolidation is cleared.
vCellReference	Set this value to <code>TM1ArrayNull()</code> . This argument is not used for clear spread.
sSpreadData	The spread code for clear spread is C. Refer to "Spreading control codes" on page 410 for complete information.

Percent change

The percent change method multiplies current cell values by a specified percentage. The product of that multiplication can then replace, be added to, or subtracted from existing cell values.

To call `TM1CubeCellSpreadViewArray`, and execute a percent change function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with <code>TM1ValPoolCreate</code> .

Argument	Description
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect.
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
aCellRange	This argument is a TM1V containing an array of row and column pairs. There is either one pair of integers, or two pairs of integers. If the array contains one pair of integers, the intersection of the row and column pair is the starting cell for the spread. For example, suppose the array was constructed as follows: <pre>aCellRange = {Row, Column} Row = TM1ValIndex(hPool,1); Column = TM1ValIndex(hpool, 1);</pre> This indicates that the starting point for the spread is the top left-hand corner of the view. This location can be either a consolidated cell or a leaf cell.
	If the array contains two pairs of integers, the two pairs define a range of values. For example, the aCellRange array for a proportional spread could be constructed as follows: <pre>aCellRange = {Row1, Column1, Row2, Column2}; Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool, 3); Row = TM1ValIndex(hPool, 3); Column = TM1ValIndex(hpool, 4);</pre>
aCellRef	Set this value to TM1ArrayNull(). This argument is not used for a percent change spread function.
sControl	The spread code for percent change is P%. If you want to add to the total already in the cell, be sure to include the + sign in your command string. Refer to “Spreading control codes” on page 410 for complete information. Directional information is not required when aCellRange defines a range of cells. It is optional if aCellRange defines a single cell.

To call `TM1CubeCellSpread`, and execute a percent change spread function, the arguments must be set as described in the following table.

Argument	Description
<code>hPool</code>	<code>hPool</code> is a pool handle obtained with <code>TM1ValPoolCreate</code> .
<code>hServer</code>	<code>hServer</code> is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
<code>vArrayOfCells</code>	<code>vArrayOfCells</code> is a TM1V containing an array cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1, Array2, Array3..., Arrayn}; Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle}; Array2 = {CubeHandle2, ElementHandle, ElementHandle..., ElementHandle};</pre> The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property <code>TM1DimensionElements</code>.
<code>vCellReference</code>	Set this value to <code>TM1ArrayNull()</code> . This argument is not used for clear spread.
<code>sSpreadData</code>	The spread code for percent change is <code>P%</code>. If you want to add to the total already in the cell, be sure to include the <code>+</code> sign in your command string. Refer to “Spreading control codes” on page 410 for complete information.

Straight line

The straight line data spreading method populates cube cells by linear interpolation between two specified endpoints.

To call `TM1CubeCellSpreadViewArray`, and execute a Straight Line function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect.
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
aCellRange	This argument is a TM1V containing an array of row and column pairs. There is either one pair of integers, or two pairs of integers. If the array contains one pair of integers, the intersection of the row and column pair is the starting cell for the spread. For example, suppose the array was constructed as follows: <pre>aCellRange = {Row, Column} Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool,1);</pre> This indicates that the starting point for the spread is the top left-hand corner of the view. This location can be either a consolidated cell or a leaf cell.
	If the array contains two pairs of integers, the two pairs define a range of values. For the Straight Line function, the values must appear in a straight line in the view. For example, the aCellRange array for a proportional spread could be constructed as follows: <pre>aCellRange = {Row1, Column1, Row2, Column2}; Row = TM1ValIndex(hPool, 1);Column = TM1ValIndex(hpool, 4); Row = TM1ValIndex(hPool, 3); Column = TM1ValIndex(hpool, 4);</pre>
aCellRef	Set this value to TM1ArrayNull(). This argument is not used for a straight line spread function.

Argument	Description
sControl	The spread code for straight line is SL. Refer to "Spreading control codes" on page 410 for complete information. Directional information is required when aCellRange defines a single cell. It must not be included when aCellRange defines a range of cells.

To call TM1CubeCellSpread, and execute a straight line spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to a TM1 server object. This handle is returned by the function TM1SystemServerConnect.
vArrayOfCells	vArrayOfCells is a TM1V containing an array cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1, Array2, Array3..., Arrayn}; Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle}; Array2 = {CubeHandle2, ElementHandle, ElementHandle..., ElementHandle};</pre> The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property TM1DimensionElements.
vCellReference	Set this value to TM1ArrayNull(). This argument is not used for straight line.
sSpreadData	The spread code for straight line is SL. Refer to "Spreading control codes" on page 410 for complete information. Directional information is required.

Growth%

The growth % method accepts an initial value and a growth percentage. Using the initial value as a starting point, this method then sequentially increments all values in a range by the specified growth percentage.

To call `TM1CubeCellSpreadViewArray`, and execute a `Growth%` function, the arguments must be set as described in the following table.

Argument	Description
<code>hPool</code>	<code>hPool</code> is a pool handle obtained with <code>TM1ValPoolCreate</code> .
<code>hServer</code>	<code>hServer</code> is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
<code>hView</code>	<code>hView</code> is a handle to a view. A handle to a new view is returned by the function <code>TM1ViewCreate</code> . Handles to existing views are stored in the cube list property <code>TM1CubeViews</code> .
<code>aCellRange</code>	This argument is a <code>TM1V</code> containing an array of row and column pairs. There is either one pair of integers, or two pairs of integers. If the array contains one pair of integers, the intersection of the row and column pair is the starting cell for the spread. For example, suppose the array was constructed as follows: <pre>aCellRange = {Row, Column} Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool, 1);</pre> This indicates that the starting point for the spread is the top left-hand corner of the view. This location can be either a consolidated cell or a leaf cell.
	If the array contains two pairs of integers, the two pairs define a range of values. For the <code>Growth%</code> function, the values must appear in a straight line in the view. For example, the <code>aCellRange</code> array for a proportional spread could be constructed as follows: <pre>aCellRange = {Row1, Column1, Row2, Column2}; Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool, 4); Row = TM1ValIndex(hPool, 3); Column = TM1ValIndex(hpool, 4);</pre>
<code>aCellRef</code>	Set this value to <code>TM1ArrayNull()</code> . This argument is not used for a <code>growth%</code> spread function.

Argument	Description
sControl	The spread code for Growth% is G%. Refer to "Spreading control codes" on page 410 for complete information. Directional information is required when aCellRange defines a single cell.

To call TM1CubeCellSpread, and execute a Growth% spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect.
vArrayOfCells	vArrayOfCells is a TM1V containing an array cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1, Array2, Array3..., Arrayn}; Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle}; Array2 = {CubeHandle2, ElementHandle, ElementHandle..., ElementHandle};</pre> The cube handles can refer to different cubes. This allows you to spread data to multiple cubes with a single spreading command. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the IBM Cognos TM1 dimension list property TM1DimensionElements.
vCellReference	Set this value to TM1ArrayNull(). This argument is not used for Growth%.
sSpreadData	The spread code for Growth% is G%. Refer to "Spreading control codes" on page 410 for complete information. Directional information is required when aCellRange defines a single cell.

Relative proportional spread

The relative proportional spread method spreads values to the leaves (children) of a consolidation proportional to the leaves of a reference cell.

The reference cell can be located in the cube from which you initiate spreading or in a separate cube. The reference cell must, however, share the same exact consolidations as the cell from which you initiate spreading.

To call `TM1CubeCellSpreadViewArray`, and execute a relative proportional spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
hView	hView is a handle to a view. A handle to a new view is returned by the function <code>TM1ViewCreate</code> . Handles to existing views are stored in the cube list property <code>TM1CubeViews</code> .
aCellRange	This argument is a TM1V containing a row and column pair. For relative proportional spread, only one pair of integers is required. This row and column pair must define a single cell consolidation.
aCellRef	aCellRef is a TM1V containing a TM1 array. aCellRef is the reference cell for aCellRange. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. aCellRef may refer to a cell in any cube. This cell must be a consolidated cell. All the consolidations in the target cell must be present in the reference cell.
sControl	The spread code for relative proportional spread is RP. Refer to "Spreading control codes" on page 410 for complete information.

To call `TM1CubeCellSpread`, and execute a relative proportional spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	hServer is a handle to a TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .

Argument	Description
vArrayOfCells	vArrayOfCells is a TM1V containing an array of cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1};Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle};</pre> For relative proportional spread, only a single cell reference is used. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property TM1DimensionElements.
vCellReference	vCellReference is a TM1V containing a TM1 array. vCellReference is the reference cell for vArrayOfCells. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. Both vCellReference and vArrayOfCells must be single cell ranges. vCellReference may refer to a cell in any cube, but the target cell must be a consolidated cell, and the consolidation must be identical to the one referenced by vArrayOfCells.
sSpreadData	The spread code for relative proportional spread is RP. Refer to "Spreading control codes" on page 410 for complete information.

Relative percent adjustment

To call TM1CubeCellSpreadViewArray, and execute a relative percent adjustment function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect.
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.

Argument	Description
aCellRange	This argument is a TM1V containing a row and column pair. For relative percent adjustment, only one pair of integers is required. This row and column pair must define a single cell consolidation. The intersection of the row and column pair is the starting cell for the spread.
aCellRef	aCellRef is a TM1V containing a TM1 array. aCellRef is the reference cell for aCellRange. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. aCellRef may refer to a cell in any cube. This cell must be a consolidated cell. All the consolidations in the target cell must be present in the reference cell.
sControl	The spread code for relative percent adjustment is R%. Refer to "Spreading control codes" on page 410 for complete information.

To call TM1CubeCellSpread, and execute a relative proportional spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to a TM1 server object. This handle is returned by the function TM1SystemServerConnect.
vArrayOfCells	vArrayOfCells is a TM1V containing an array of cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1}; Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle};</pre> For relative percent adjustment, only a single cell reference is used. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property TM1DimensionElements.

Argument	Description
vCellReference	<i>vCellReference</i> is a TM1V containing a TM1 array. <i>vCellReference</i> is the reference cell for <i>vArrayOfCells</i>. The array contains the reference cell's cube handle, and a set of element handles that identifies the cell. Both <i>vCellReference</i> and <i>vArrayOfCells</i> must be single cell ranges. <i>vCellReference</i> may refer to a cell in any cube, but the target cell must be a consolidated cell, and the consolidation must be identical to the one referenced by <i>vArrayOfCells</i>.
sSpreadData	The spread code for relative percent adjustment is R%. Refer to "Spreading control codes" on page 410 for complete information.

Repeat leaves

To call `TM1CubeCellSpreadViewArray`, and execute a repeat leaves function, the arguments must be set as described in the following table.

Argument	Description
hPool	<i>hPool</i> is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	<i>hServer</i> is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
hView	<i>hView</i> is a handle to a view. A handle to a new view is returned by the function <code>TM1ViewCreate</code> . Handles to existing views are stored in the cube list property <code>TM1CubeViews</code> .
aCellRange	This argument is a TM1V containing a row and column pair. For repeat leaves, only one pair of integers is required. This row and column pair must define a single cell consolidation. The intersection of the row and column pair is the starting cell for the spread.
aCellRef	Set this value to <code>TM1ArrayNull()</code> . This argument is not used for a repeat leaves spread function.
sControl	The spread code for repeat leaves is LR. Refer to "Spreading control codes" on page 410 for complete information.

To call `TM1CubeCellSpread`, and execute a repeat leaves spread function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	hServer is a handle to a TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .
vArrayOfCells	vArrayOfCells is a TM1V containing an array of cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1}; Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle};</pre> For repeat leaves, only a single cell reference is used. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property <code>TM1DimensionElements</code>.
vCellReference	Set this value to <code>TM1ArrayNull()</code> . This argument is not used for repeat leaves.
sSpreadData	The spread code for repeat leaves is LR. Refer to "Spreading control codes" on page 410 for complete information.

Equal spread leaves

To call `TM1CubeCellSpreadViewArray`, and execute an equal spread leaves function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with <code>TM1ValPoolCreate</code> .
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function <code>TM1SystemServerConnect</code> .

Argument	Description
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.
aCellRange	This argument is a TM1V containing a row and column pair. For equal spread leaves, only one pair of integers is required. This row and column pair must define a single cell consolidation. The intersection of the row and column pair is the starting cell for the spread.
aCellRef	Set this value to TM1ArrayNull(). This argument is not used for a equal spread leaves spread function.
sControl	The spread code for equal spread leaves is LS. Refer to "Spreading control codes" on page 410 for complete information.

To call TM1CubeCellSpread, and execute a equal spread leaves function, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to a TM1 server object. This handle is returned by the function TM1SystemServerConnect.
vArrayOfCells	vArrayOfCells is a TM1V containing an array of cell references. This array of cell references takes the form: <pre>VArrayOfCells = {Array1};Array1 = {CubeHandle1, ElementHandle, ElementHandle..., ElementHandle};</pre> For equal spread leaves, only a single cell reference is used. The starting location for the spread is the intersection of the Element Handles within the specified cube. Element handles are extracted from the TM1 dimension list property TM1DimensionElements.

Argument	Description
vCellReference	Set this value to TM1ArrayNull(). This argument is not used for equal spread leaves.
sSpreadData	The spread code for repeat leaves is LS. Refer to “Spreading control codes” on page 410 for complete information.

Applying holds

Applying and releasing holds on leaf cells and consolidations is done through the same API functions you use to spread data: TM1CubeCellSpreadViewArray and TM1CubeCellSpread.

The arguments you pass to these functions to apply or release holds are very similar. There are special spreading commands for holding leaves and consolidations, and for releasing holds on leaves and consolidations.

This section describes the arguments required to hold a set of leaf cells in a view. Other hold commands are very similar to this example.

To call TM1CubeCellSpreadViewArray, and apply holds to a set of leaf cells, the arguments must be set as described in the following table.

Argument	Description
hPool	hPool is a pool handle obtained with TM1ValPoolCreate.
hServer	hServer is a handle to an IBM Cognos TM1 server object. This handle is returned by the function TM1SystemServerConnect.
hView	hView is a handle to a view. A handle to a new view is returned by the function TM1ViewCreate. Handles to existing views are stored in the cube list property TM1CubeViews.

Argument	Description
aCellRange	This argument is a TM1V containing an array of row and column pairs. There is either one pair of integers, or two pairs of integers. If the array contains one pair of integers, the intersection of the row and column pair is the target cell for the spread command. For example, suppose the array was constructed as follows: <pre>aCellRange = {Row, Column} Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool, 1);</pre> This indicates that the starting point for the spread is the top left-hand corner of the view. For a leaf hold command, this location must be a leaf cell.
	If the array contains two pairs of integers, the two pairs define a range of values. For example, the aCellRange array for a proportional spread could be constructed as follows: <pre>aCellRange = {Row1, Column1, Row2, Column2}; Row = TM1ValIndex(hPool, 1); Column = TM1ValIndex(hpool, 3); Row = TM1ValIndex(hPool, 3); Column = TM1ValIndex(hpool, 4);</pre>
aCellRef	Set this value to TM1ArrayNull(). This argument is not used for hold functions.
sControl	The spread code for a leaf hold function is H. The spread code for a consolidated hold function is HC. Refer to “Spreading control codes” for complete information.

Spreading control codes

The following table provides information about the spreading control code argument in the TM1CubeCellSpreadViewArray and TM1CubeCellSpread functions.

This code is a string built from the elements described in the following table. Note that the Update Action parameter is optional, as described in Note 1.

Data Spreading Method	Code / Update Action / Direction Indicators / Required Method Parameters	Example
Proportional Spread	P +, ~ , ^, <, > Value to be spread	P<>100 This example proportionally spreads the value 100 to all leaf cells on the row of insertion, replacing existing cell values.
Equal Spread	S +, ~ , ^, <, > Value to be spread	S+ ^200 This example equally spreads the value 200 to all leaf cells on the column of insertion, adding the product of spreading to existing cell values.
Repeat	R +, ~ , ^, <, > Value to be spread	R~<50 This example subtracts the value 50 from all leaf cells left of the insertion point.
Percent Change	P% +, ~ , ^, <, > Percentage	P%+ ^<>10 This example applies a percent change of 10% to all leaf values in the view and adds the product to existing cell values.(It increments all leaves in the view by 10%.)
Straight Line	SL +, ~ , ^, <, > ∞ Start Value and End Value	SL>100:200 This example applies Straight Line spreading to replace all leaf values right of the point of insertion, using a start value of 100 and an end value of 200.
Growth %	GR +, ~ , ^, <, > ∞ Start Value and Growth Percentage	GR ^300:25 This example applies a 25% growth percentage to the starting value of 300 and replaces all leaf values below the point of insertion.

Data Spreading Method	Code / Update Action / Direction Indicators / Required Method Parameters	Example
Clear	C N/A , ^, <, > N/A	C ^ < > This example clears values from all cells in the view.
Relative Proportional Spread	RP +, ~ N/A Value to be spread	RP+500 Distributes the value 500 proportionally across all leaf cells of the consolidation.
Relative Percent adjustment	R% +, ~ N/A Percentage adjustment	R%+20 For each leaf cell in the consolidation, take the value of the reference cell's analogous leaf cell, add 20%, and add the total to the leaf cell in the current consolidation.
Repeat Leaves	LR +, ~ The default is to update populated cells. * means update all leaf cells Value to be spread	LR+*200 Add 200 to all populated leaves in the current consolidation.
Equal spread Leaves	LS +, ~ The default is to update populated cells. * means update all leaf cells Value to be spread	LS+*200 Spread 200 to all leaves in the current consolidation.

Data Spreading Method	Code / Update Action / Direction Indicators / Required Method Parameters	Example
Leaf Hold	H N/A , ^, <, > N/A	H<> This example holds all leaf cells on the row of insertion.
Release Leaf Hold	RH N/A , ^, <, > N/A	RH<> This example releases all leaf holds on the row of insertion.
Consolidation Hold	HC N/A , ^, <, > N/A	HC<> This example holds all consolidated cells on the row of insertion.
Release Consolidation Hold	RC N/A , ^, <, > N/A	RC<> This example releases all consolidated cells on the row of insertion.
1. The default data action is Replace. The spreading syntax uses the tilde character (~) to denote the Subtract data action, and the plus symbol (+) to denote the Add data action. 2. Straight Line and Growth % methods can be used across a single row or column. Rectangular ranges are not allowed. Direction combinations of up and down (^) or left and right(<>) are the only combinations allowed for these spreading methods.		

Notices

This information was developed for products and services offered worldwide.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Software Group
Attention: Licensing
3755 Riverside Dr
Ottawa, ON K1V 1B7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, TM1, Express, and Cognos are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “ Copyright and trademark information ” at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies:

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.

Index

Special characters

-Z switch 2

A

- access rights
 - determining 16
 - read, C 198
 - read, Visual Basic 340
 - write, C 199
 - write, Visual Basic 340
- accessing log files 20
- Admin
 - group 15
 - privileges 16
 - server 3
 - server, tmladmnserv.exe 3
- Admin Host 2, 8
 - setting location of 8
 - setting name, C 177
 - setting name, Visual Basic 320, 321
- administration
 - TM1 security 14
- arrays 12
 - component, retrieving 190, 330, 331
 - component, retrieving number of 191, 332
 - component, updating 191, 332
 - establishing value, C 192
 - establishing value, Visual Basic 333
 - handles 388
 - values functions, C 192
 - values functions, Visual Basic 334
- attributes
 - delete from object, C 114
 - delete from object, Visual Basic 257
 - insert in object, C 115
 - insert in object, Visual Basic 258
 - object 14
 - types 359
 - values, retrieving 116, 259
 - values, setting 116, 260, 261
- authentication 5

B

- backup
 - and recovery 18
 - functions 5
- BLOB
 - as child object 51
 - closing, C 56
 - closing, Visual Basic 227
 - create 51
 - create, C 56
 - create, Visual Basic 227
 - delete 52
 - files 52
 - list 52
 - objects 51
 - opening, C 57

- BLOB (*continued*)
 - opening, Visual Basic 229
 - properties 51, 52
 - registering 51
 - retrieving data, C 57
 - retrieving data, Visual Basic 228
 - update 51
 - writing to, C 58
 - writing to, Visual Basic 229

C

- C/C++
 - .dll files 21
 - build, run program 22
 - creating project with Microsoft Visual Studio 21
 - setting paths in Microsoft Visual C++ 21
 - test code 22
- ChangeSet 19
- child
 - Blob 51
 - of cubes 39
- clients 3
 - access rights 14, 16
 - adding 15
 - assigning to group 16
 - assigning to group, C 63
 - assigning to group, Visual Basic 232
 - deleting 15
 - disconnecting 31
 - password, C 65
 - password, Visual Basic 234
 - removing from group, C 64
 - removing from group, Visual Basic 233
- code sample
 - C 5
 - Visual Basic 7
- compilers 2
- connecting to the API 5
- constants object types 374
- conventions
 - API 1
- create
 - BLOB 51
 - BLOB, C 56
 - subset 44
- cube
 - as child objects 39
 - as parent object 39
 - attaching rule to, C 139
 - attaching rule to, Visual Basic 286
 - creating, C 86, 218
 - creating, handle 40
 - creating, Visual Basic 246, 355
 - definition 39
 - detaching rule from, C 141
 - detaching rule from, Visual Basic 288
 - dimensions 39
 - input 39
 - objects 53
 - objects defined 53

- cube *(continued)*
 - perspectives 41
 - properties 40
 - retrieving value, C 83
 - retrieving value, Visual Basic 245
 - shared 18
 - updating cell 40
 - updating cell, C 84
 - updating cell, Visual Basic 245
 - views 365

D

- data recovering 20
- Data Reservation API functions
 - error codes 17
 - TM1DataReservationAcquire 91
 - TM1DataReservationGetAll 92
 - TM1DataReservationGetConflicts 93
 - TM1DataReservationRelease 94
 - TM1DataReservationReleaseAll 95
 - TM1DataReservationValidate 95
- data spreading syntax
 - C 222
 - Visual Basic 239
- data types
 - handles 9
 - values 10
- delete
 - subset 44
- dimensions
 - adding elements to, C 98
 - adding elements to, Visual Basic 251
 - as parent objects 36, 43
 - checking consistency, C 97
 - checking consistency, Visual Basic 250
 - creating 36
 - creating, C 96
 - creating, Visual Basic 250
 - deleting 37
 - deleting subsets 367
 - inserting element, C 101
 - inserting element, Visual Basic 254
 - list 38
 - objects 53
 - objects defined 53
 - properties list 38
 - registering 36
 - replacing, C 108, 176
 - replacing, Visual Basic 256, 319
 - updating 20, 37
- disconnect 8
 - client 31

E

- elements
 - adding dimension to, C 98
 - adding dimension to, Visual Basic 251
 - consolidation hierarchies 37
 - deleting consolidated, C 99
 - deleting consolidated, Visual Basic 252
 - dimension, C 101
 - dimension, Visual Basic 254
 - retrieving component weight, C 100
 - retrieving component weight, Visual Basic 253

- elements *(continued)*
 - sorting, C 175
 - sorting, Visual Basic 318
 - types 37

- Ell
 - C 158
 - Visual Basic 302

- error codes
 - extracting 11
 - extracting from value, C 194
 - extracting from value, Visual Basic 335
 - naming convention 17

F

- files
 - deleting object files, C 120
 - deleting object files, Visual Basic 265
 - loading object files, C 120
 - loading object files, Visual Basic 265
- functions
 - BLOB list 52
 - dimension list 38
 - naming conventions 53
 - object 53
 - object, C 121
 - object, creation 31
 - object, defined 53
 - object, list 29
 - private object 31
 - private object, C 127
 - private object, Visual Basic 273
 - rule 42
 - server 3, 35
 - server, objects 53
 - subset 45
 - system 53
 - types 53
 - ValBool, C 192
 - ValBool, Visual Basic 334
 - value capsule 53
 - view 50

G

- group rights
 - retrieving, C 133
 - retrieving, Visual Basic 280
 - setting, C 134
 - setting, Visual Basic 281
- groups
 - adding to server, C 113
 - adding to server, Visual Basic 257
 - ADMIN 15
 - assigning, clients to 16, 63, 232
 - assigning, rights 16
 - changing access 16
 - functions, C 62
 - functions, Visual Basic 231
 - removing clients from, C 64
 - removing clients from, Visual Basic 233

H

- handles
 - cube views 365

- handles (*continued*)
 - data types 9
 - element 366, 388
- header
 - file for applications 1
- hierarchies 37
 - drawing, C 157
 - drawing, Visual Basic 301
 - sorting by, C 175
 - sorting by, Visual Basic 318

I

- index value capsule
 - retrieving, C 195
 - retrieving, Visual Basic 337
 - updating, C 196
 - updating, Visual Basic 338
- initialize 5
- integrated login 5
- IntegratedSecurityMode 5

L

- link libraries 1
- list properties
 - retrieving by index, C 122
 - retrieving by index, Visual Basic 268
 - retrieving by name, C 123
 - retrieving by name, Visual Basic 269
 - retrieving number of items, C 122, 124
 - retrieving number of items, Visual Basic 267, 270
 - returns handle, C 125
 - returns handle, Visual Basic 271
- LoadLibrary function 1
- locks 4
- log files
 - accessing 20
 - accessing, C 148
 - accessing, Visual Basic 295
 - purpose 30
 - retrieving next data item, C 147
 - retrieving next data item, Visual Basic 294
 - terminating access, C 147
 - terminating access, Visual Basic 293
- logical object 2

M

- meta security 16
- metadata 4
- Microsoft Visual Basic 327, 349
 - support 1
 - version supported 2
- Microsoft Visual C++ 2
 - header files 21
- multiple readers 4
- multiprocessor 4

N

- naming conventions 53
- network address 3

O

- objects
 - access 16
 - access rights 16
 - assigning rights to 16
 - attributes 14
 - copying, C 117
 - copying, Visual Basic 262
 - deleting, C 118
 - deleting, dependencies 33
 - deleting, files 120, 265
 - deleting, private 124, 270
 - deleting, Visual Basic 263
 - destroying 33
 - destroying, C 118
 - destroying, Visual Basic 263
 - duplicating, C 119
 - duplicating, Visual Basic 264
 - functions list 29
 - handles 27
 - loading 33
 - loading object files, C 120
 - names 32
 - navigating 29
 - parent 29
 - private 31
 - private, C 127
 - private, list property 124, 270, 271
 - private, publish 127, 273
 - private, register 127, 273
 - private, Visual Basic 273
 - properties 16, 28
 - public 31
 - registering 30
 - registering, C 130
 - registering, Visual Basic 276
 - retrieving, type 200, 342
 - retrieving, value capsule contents 199, 341
 - retrieving, value of property 128, 274
 - returning value 12
 - storage 12, 30
 - type 12, 27
 - updating, property value 129, 275
 - updating, value capsule 200, 342
- order
 - define elements C 101
 - library load order 1

P

- packet size 3
- parent objects 29, 36, 39, 43
 - BLOB 51
- parent subset 43
- passwords
 - C 65, 150
 - Visual Basic 234, 297
- PATH 1
- pending value 13
- performance 4
- perspectives
 - calculating 41
 - creating 41
 - creating, C 88
 - creating, Visual Basic 247
 - destroying 41

- perspectives (*continued*)
 - destroying, C 90
 - destroying, Visual Basic 248
 - storing permanently 41
- pool handles 8
- Port number 3
- private objects
 - Visual Basic 127, 273
- progress callback functions 9
 - setting 9
- progress messages
 - C 180
- properties
 - access rights 16
 - BLOB 52
 - cubes 40
 - dimension 38
 - list 28
 - rule 42
 - server 34
 - TM1AttributeType 359
 - TM1BlobSize 359
 - TM1ClientPassword 360
 - TM1ClientStatus 360
 - TM1ConnectionChoresUsing 360
 - TM1ConnectionLastSyncTime 361
 - TM1ConnectionLastSyncTimeStar 361
 - TM1ConnectionSyncErrorCount 362
 - TM1ConnectionSyncPlanetToStar 362
 - TM1ConnectionSyncStarToPlanet 362
 - TM1ConnectionUsername 361
 - TM1CubeCellValueUndefined 363
 - TM1CubeDimensions 363
 - TM1CubeMeasuresDimension 363
 - TM1CubePerspectivesMaxMemory 364
 - TM1CubePerspectivesMinTime 364
 - TM1CubeReplicationSyncRule 364
 - TM1CubeReplicationSyncViews 365
 - TM1CubeRule 365
 - TM1CubeTimeDimension 365
 - TM1CubeViews 365
 - TM1DimensionCubesUsing 366
 - TM1DimensionElements 366
 - TM1DimensionNofLevels 366
 - TM1DimensionReplicationSyncSubsets 367
 - TM1DimensionSubsets 367
 - TM1DimensionTopElement 368
 - TM1DimensionWidth 368
 - TM1ElementComponents 368
 - TM1ElementIndex 369
 - TM1ElementLevel 369
 - TM1ElementParents 369
 - TM1ElementType 370
 - TM1ObjectAttributes 370
 - TM1ObjectChangedSinceLoaded 370
 - TM1ObjectLastTimeUpdated 371
 - TM1ObjectMemoryUsed 371
 - TM1ObjectName 371
 - TM1ObjectNull 372
 - TM1ObjectParent 372
 - TM1ObjectRegistration 372
 - TM1ObjectReplicationConnection 373
 - TM1ObjectReplicationSourceObjectName 373
 - TM1ObjectReplicationStatus 373
 - TM1ObjectSecurityOwner 374
 - TM1ObjectSecurityStatus 374
 - TM1ObjectType 374

- properties (*continued*)
 - TM1RuleErrorLine 375
 - TM1RuleErrorString 375
 - TM1RuleNofLines 376
 - TM1ServerBlobs 376
 - TM1ServerBuildNumber 376
 - TM1ServerChores 377
 - TM1ServerClients 377
 - TM1ServerConnections 377
 - TM1ServerCubes 377
 - TM1ServerDimensions 378
 - TM1ServerDirectories 378
 - TM1ServerGroups 378
 - TM1ServerLogDirectory 378
 - TM1ServerNetworkAddress 379
 - TM1ServerProcesses 379
 - TM1SQLTableColumnNameNames 379
 - TM1SQLTableColumnTypeNames 379
 - TM1SQLTableNumberOfColumns 380
 - TM1SQLTableNumberOfRows 380
 - TM1SubsetAlias 381
 - TM1SubsetElements 381
 - TM1SubsetExpression 381
 - TM1ViewColumnSubsets 382
 - TM1ViewExtractComparison 382
 - TM1ViewExtractRealLimitA 382
 - TM1ViewExtractRealLimitB 383
 - TM1ViewExtractSkipConsolidatedValues 383
 - TM1ViewExtractSkipRuleValues 383
 - TM1ViewExtractSkipZeroes 383
 - TM1ViewExtractStringLimitA 384
 - TM1ViewExtractStringLimitB 384
 - TM1ViewFormat 384
 - TM1ViewFormatString 385
 - TM1ViewPreConstruct 386
 - TM1ViewRowSubsets 387
 - TM1ViewShowAutomatically 387
 - TM1ViewSuppressZeroes 387
 - TM1ViewTitleElements 388
 - TM1ViewTitleSubsets 388
 - updating, C 129
 - updating, Visual Basic 275
- property object
 - BLOB 51
- protocol 3

R

- READ 2
- recovering data 20
- recovery
 - backup 18
 - backup function 5
 - cubes 18
 - restore function 5
- register
 - objects list 31
 - subset 44
- registering objects
 - BLOB 51
 - cubes 39
 - defined 30
 - dimensions 36
 - private 44
 - private, C 127
 - private, Visual Basic 273
 - subsets 44

- registration
 - BLOB 51
- remote server 2
- restoring
 - cubes 18
 - function 5
- retrieving
 - call result, C 62
 - call result, Visual Basic 231
 - handle, C 62
 - handle, Visual Basic 231
- rules 42
 - adding lines 43
 - as child objects 39
 - attaching to cube 43
 - attaching to cube, C 139
 - attaching to cube, Visual Basic 286
 - checking syntax 43
 - checking syntax, C 139
 - checking syntax, Visual Basic 287
 - copying 43
 - creating 43
 - creating, C 140
 - creating, Visual Basic 288
 - deleting lines 43
 - detaching from cube, C 141
 - detaching from cube, Visual Basic 288
 - file format 42
 - inserting line, C 142
 - inserting line, Visual Basic 290
 - inserting statements 43
 - properties 42
 - removing 43
 - retrieving line, C 141
 - retrieving line, Visual Basic 289
 - updating 43

S

- samples 5
- security
 - assigning rights 16
 - levels 14
 - locking, C 131
 - locking, Visual Basic 278
 - meta security 16
 - releasing, C 132
 - releasing, Visual Basic 279
 - reserving, C 132
 - reserving, Visual Basic 279
 - rights, retrieving 133, 280
 - rights, returning 16
 - rights, setting 134, 281
 - unlocking, C 135
 - unlocking, Visual Basic 282
- servers
 - as parent object 51
 - disconnecting, C 183
 - disconnecting, Visual Basic 326
 - functions 3, 35
 - log file, accessing 148, 295
 - log file, terminating access 147, 293
 - objects 53
 - objects, as parent 36, 39
 - objects, creating 34
 - objects, destroying 34
 - objects, properties 34

- servers (*continued*)
 - objects, retrieving 184, 326
 - retrieving, by name 185
 - retrieving, number available 185, 328
 - starting, local 2
 - starting, remote 2
 - starting, with function call 186, 187, 329
 - stopping, C 188
 - stopping, Visual Basic 329
 - tests, if updatable 197, 339
 - tests, type 196, 338
- sorting elements
 - C 175
 - Visual Basic 318
- spreading control code 410
 - C 222
- SSL
 - configuring clients, C 53
- storage
 - BLOB 52
- storing objects 30
- strings
 - maximum length 12
 - returning, size 11
 - returning, text 11
- subsets 43
 - adding to dimension 367
 - as child objects 36, 43
 - attribute, C 168
 - attribute, Visual Basic 310
 - creating 44
 - creating, C 157
 - creating, Visual Basic 301
 - deleting selected elements, C 171
 - deleting selected elements, Visual Basic 313
 - display, children 160, 304
 - display, level 159, 303
 - display, weight 162, 307
 - drawing tree hierarchy, C 157
 - drawing tree hierarchy, Visual Basic 301
 - element display tree 47
 - Ell, C 158
 - Ell, Visual Basic 302
 - functions 45
 - inserting, children 172, 314
 - inserting, element 165, 307
 - inserting, into subset 165, 308
 - inserting, parents 173, 315
 - line, C 159
 - line, Visual Basic 304
 - populating, C 155
 - populating, Visual Basic 299
 - removing elements from, C 173, 176
 - removing elements from, Visual Basic 316, 318
 - select, by index 169, 311
 - select, by level 170, 311
 - select, by pattern 170, 312
 - selected, C 161
 - selected, Visual Basic 306
 - sorting elements alphabetically, C 174
 - sorting elements alphabetically, Visual Basic 317
 - tee, C 162
 - tee, Visual Basic 306
- supported compilers 2
- system administration functions
 - C 189
 - Visual Basic 330

- system administration 14
- system functions
 - Admin Host server 8
 - disconnecting server 8
 - return, servers 8
 - software revision 9

T

- thread contention 4
- threads 9
- TM1 Admin Host 8
- TM1 Admin Server 3
- TM1 API
 - connecting, C 183
 - connecting, multiple 9
 - connecting, server 8
 - connecting, Visual Basic 324, 325
 - cubes object functions 53
 - cubes object functions, Visual Basic 225
 - dimension functions 53
 - disconnecting 8, 178
 - disconnecting, Visual Basic 322
 - function names, Visual Basic 225
 - object functions, Visual Basic 225
 - server object functions 53
 - server object functions, Visual Basic 225
 - system functions, Visual Basic 225
 - value capsule functions, Visual Basic 225
 - version 9
 - version, C 189
 - version, Visual Basic 330
- TM1 security 14
- TM1 server log file 30
- TM1.H, data types 9
- tm1admsrv.exe 3
- Tm1admsrv.exe 8
- TM1API.dll 1
- TM1API.H 1
- TM1APIFinalize
 - C 55
 - Visual Basic 226
- TM1APIInitialize
 - C 55
 - Visual Basic 225
- TM1AttributeType 359
- TM1BlobClose
 - C 56
 - Visual Basic 227
- TM1BlobCreate
 - Visual Basic 227
- TM1BlobGet
 - C 57
 - Visual Basic 228
- TM1BlobOpen
 - C 57
 - Visual Basic 229
- TM1BlobPut
 - C 58
 - Visual Basic 229
- TM1BlobSize 359
- TM1CancelClientJob
 - C 59
- TM1ChangeSetBegin
 - C 59
- TM1ChangeSetEnd
 - C 60

- TM1ChangeSetUndo
 - C 61
- TM1ChoreExecute
 - C 61
 - Visual Basic 230
- TM1ClientAdd
 - C 62
 - clients 15
 - Visual Basic 230
- TM1ClientGroupAssign
 - C 62
 - Visual Basic 231
- TM1ClientGroupIsAssigned
 - C 63
 - clients to groups 16
 - Visual Basic 232
- TM1ClientGroupRemove
 - C 64
 - clients to groups 16
 - Visual Basic 233
- TM1ClientHasHolds
 - C 64
 - Visual Basic 233
- TM1ClientPassword 360
- TM1ClientPasswordAssign
 - C 65
 - Visual Basic 234
- TM1ClientStatus 360
- TM1ConnectionCheck
 - C 66
 - Visual Basic 235
- TM1ConnectionChoresUsing 360
- TM1ConnectionCreate
 - C 66
 - Visual Basic 235
- TM1ConnectionDelete
 - C 67
 - Visual Basic 236
- TM1ConnectionLastSyncTime 361
- TM1ConnectionLastSyncTimeStar 361
- TM1ConnectionSyncErrorCount 362
- TM1ConnectionSynchronize
 - C 68
 - Visual Basic 237
- TM1ConnectionSyncPlanetToStar 362
- TM1ConnectionSyncStarToPlanet 362
- TM1ConnectionUsername 361
- TM1CubeCellDrillListGet
 - C 68
 - Visual Basic 237
- TM1CubeCellDrillObjectBuild
 - C 69
 - Visual Basic 238
- TM1CubeCellPickListExists
 - C 72
- TM1CubeCellPickListGet
 - C 70
- TM1CubeCellsPickListGet
 - C 71
- TM1CubeCellSpread
 - C 77
 - Visual Basic 239
- TM1CubeCellSpreadStatusGet
 - C 81
 - Visual Basic 241
- TM1CubeCellSpreadViewArray
 - C 72

TM1CubeCellSpreadViewArray *(continued)*
 Visual Basic 244
 TM1CubeCellsValueGet
 C 84
 TM1CubeCellsValueSet
 C 85
 TM1CubeCellValueGet
 C 83
 Visual Basic 245
 TM1CubeCellValueSet
 C 84
 cubes 40
 Visual Basic 245
 TM1CubeCellValueUndefined 363
 TM1CubeCreate
 C 86
 Visual Basic 246
 TM1CubeDimensions 363
 TM1CubeListByNamesGet
 C 87, 221
 TM1CubeListGet
 C 88, 222
 TM1CubeLogChanges 363
 TM1CubeMeasuresDimension 363
 TM1CubePerspectiveCreate
 C 88
 example 88
 new 41
 Visual Basic 247
 TM1CubePerspectiveDestroy
 C 90
 memory 41
 Visual Basic 248
 TM1CubePerspectiveMaximumMemory
 bytes 41
 TM1CubePerspectiveMinimumTime
 stored 41
 TM1CubePerspectivesMaxMemory 364
 TM1CubePerspectivesMinTime 364
 TM1CubeReplicationSyncViews 365
 TM1CubeRule 365
 TM1CubeShowsNulls
 C 90
 Visual Basic 249
 TM1CubeTimeDimension 365
 TM1CubeTimeLastInvalidated
 C 91
 TM1CubeViews 365
 TM1DataReservationAcquire 91
 TM1DataReservationGetAll 92
 TM1DataReservationGetConflicts 93
 TM1DataReservationRelease 94
 TM1DataReservationReleaseAll 95
 TM1DataReservationValidate 95
 TM1DimensionAttributesGet(
 C 96
 TM1DimensionCheck
 C 97
 creating new dimension 36
 updating dimension 37
 Visual Basic 250
 TM1DimensionCreateEmpty
 C 96
 creating new dimension 36
 Visual Basic 250
 TM1DimensionCubesUsing 366

TM1DimensionElementComponentAdd
 C 98
 creating new dimension 36
 updating dimension 37
 Visual Basic 251
 TM1DimensionElementComponentDelete
 C 99
 updating dimension 37
 Visual Basic 252
 TM1DimensionElementComponentWeightGet
 C 100
 Visual Basic 253
 TM1DimensionElementDelete
 C 101
 updating dimension 37
 Visual Basic 253
 TM1DimensionElementInsert
 C 101
 creating new dimension 36
 updating dimension 37
 Visual Basic 254
 TM1DimensionElementListByIndexGet
 C 107
 TM1DimensionElementListByNamesGet
 C 107
 TM1DimensionElements 366
 TM1DimensionNofLevels 366
 TM1DimensionReplicationSyncSubsets 367
 TM1DimensionSubsets 367
 TM1DimensionTopElement 368
 TM1DimensionUpdate
 C 108
 updating dimension 37
 Visual Basic 256
 TM1DimensionWidth 368
 TM1ElementCompomentsGet
 C 109
 TM1ElementComponents 368
 TM1ElementIndex 369
 TM1ElementLevel 369
 TM1ElementParents 369
 TM1ElementType 370
 TM1ErrorObjectHandleIsInvalid 17
 TM1ErrorSystemUserHandleIsInvalid 17
 TM1ErrorSystemValueIsInvalid 17
 TM1GetSubsetByHandle
 C 110
 TM1GetViewByHandle
 C 113
 TM1GetViewByName
 C 110
 TM1GroupAdd
 C 113
 creating new 15
 Visual Basic 257
 TM1LIB.dll 1
 TM1ObjectAttributeDelete
 C 114
 destroy 14
 Visual Basic 257
 TM1ObjectAttributeGet
 retrieve from list 14
 TM1ObjectAttributeInsert 359
 C 115
 create 14
 Visual Basic 258
 TM1ObjectAttributes 370

TM1ObjectAttributeSet	TM1ObjectPrivateListCountGet (continued)
updates 14	Visual Basic 270
TM1ObjectAttributeValueGet	TM1ObjectPrivateListHandleByIndexGet
C 116	C 125
Visual Basic 259	Visual Basic 271
TM1ObjectAttributeValueSet	TM1ObjectPrivateListHandleByNameGet
C 116	C 126
Visual Basic 260	Visual Basic 272
TM1ObjectAttributeValuesSet	TM1ObjectPrivatePublish
VB 261	C 127
TM1ObjectChangedSinceLoaded 370	Visual Basic 273
TM1ObjectCopy	TM1ObjectPrivateRegister
C 117	access 31
Visual Basic 262	C 127
TM1ObjectDelete	public 30
C 118	Visual Basic 273
clients 15	TM1ObjectPropertyGet
cube 40	C 128
deleting dimension 37	Visual Basic 274
group 15	TM1ObjectPropertySet 275
subset 44	C 129
Visual Basic 263	TM1ObjectRegister
TM1ObjectDestroy	C 130
C 118	dimension 36
cube 40	public 30, 39
deleting dimension 37	registering dimension 36
subset 44	subsets 44
Visual Basic 263	Visual Basic 276
TM1ObjectDuplicate 37	TM1ObjectRegisterPrivate
C 119	subset 44
updating 43	TM1ObjectRegistration 372
updating subset 44	TM1ObjectReplicate
Visual Basic 264	C 130
TM1ObjectFileDelete	Visual Basic 277
C 120	TM1ObjectReplicationSourceObjectName 373
Visual Basic 265	TM1ObjectReplicationStatus 373
TM1ObjectFileLoad	TM1ObjectSecurityClientRight 16
C 120	TM1ObjectSecurityIsLocked
TM1ObjectFileLoad objects	lock 16
loading object files, Visual Basic 265	TM1ObjectSecurityIsReserved
Visual Basic 265	lock 16
TM1ObjectFileSave	TM1ObjectSecurityLock
C 121	C 131
TM1ObjectFileSave functions	update 16
object, Visual Basic 266	Visual Basic 278
Visual Basic 266	TM1ObjectSecurityOwner 374
TM1ObjectLastTimeUpdated 371	TM1ObjectSecurityRelease
TM1ObjectListCountGet	C 132
C 122	locks 16
Visual Basic 267	Visual Basic 279
TM1ObjectListHandleByIndexGet	TM1ObjectSecurityReserve
C 122	C 132
Visual Basic 268	locks 16
TM1ObjectListHandleByNameGet	Visual Basic 279
C 123	TM1ObjectSecurityRightGet
Visual Basic 269	assigning rights 16
TM1ObjectMemoryUsed 371	C 133
TM1ObjectName 371	Visual Basic 280
TM1ObjectNull 372	TM1ObjectSecurityRightSet
TM1ObjectParent 372	assigning rights 16
locate 31	C 134
TM1ObjectPrivateDelete	Visual Basic 281
C 124	TM1ObjectSecurityStatus 374
subset 44	TM1ObjectSecurityUnlock
Visual Basic 270	C 135
TM1ObjectPrivateListCountGet	privileges 16
C 124	Visual Basic 282

- TM1ObjectType 374
- TM1P 5
- TM1ProcessExecute
 - C 135, 137
 - Visual Basic 283, 284
- TM1ProcessExecuteSQLQuery
 - C 138
 - Visual Basic 285
- TM1ProcessVariableNameIsValid
 - C 138
 - Visual Basic 286
- TM1RuleAttach
 - C 139
 - replacing 43
 - Visual Basic 286
- TM1RuleCheck
 - C 139
 - defining 43
 - Visual Basic 287
- TM1RuleCreateEmpty
 - C 140
 - handle 43
 - Visual Basic 288
- TM1RuleDetach
 - C 141
 - removing 43
 - Visual Basic 288
- TM1RuleErrorLine 375
- TM1RuleErrorString 375
- TM1RuleLineDelete
 - adding 43
- TM1RuleLineGet
 - C 141
 - Visual Basic 289
- TM1RuleLineInsert
 - C 142
 - deleting 43
 - Visual Basic 290
- TM1RuleNofLines 376
- TM1RulesNofLines
 - text 42
- TM1S.CFG 2
- TM1S.EXE 2
- TM1S.LOG 20
- TM1ServerBatchUpdateFinish
 - C 143
 - Visual Basic 291
- TM1ServerBatchUpdateIsActive
 - C 143
 - Visual Basic 292
- TM1ServerBatchUpdateStart
 - C 144
 - Visual Basic 292
- TM1ServerBlobs 376
- TM1ServerBuildNumber 376
- TM1ServerChores 377
- TM1ServerClients 377
- TM1ServerConnections 377
- TM1ServerCubes 377
- TM1ServerDimensionListByNamesGet
 - C 145
- TM1ServerDimensionListGet
 - C 86
- TM1ServerDimensions 378
- TM1ServerDirectories 378
- TM1ServerDisableBulkLoadMode
 - C 146
- TM1ServerEnableBulkLoadMode
 - C 146
- TM1ServerGroups 378
 - property 378
- TM1ServerLogClose
 - C 147
 - terminate access 20
 - Visual Basic 293
- TM1ServerLogDirectory 378
- TM1ServerLogNext
 - C 147
 - retrieve field 20
 - Visual Basic 294
- TM1ServerLogOpen
 - C 148
 - time certain 20
 - Visual Basic 295
- TM1ServerNetworkAddress 379
- TM1ServerOpenSQLQuery 296
 - C 149
 - Visual Basic 296
- TM1ServerPasswordChange
 - C 150
 - Visual Basic 297
- TM1ServerProcesses 379
- TM1ServerSandboxesDelete
 - C 151
- TM1ServerSecurityRefresh
 - C 154
 - Visual Basic 298
- TM1SQLTableColumnNames 379
- TM1SQLTableColumnTypes 379
- TM1SQLTableGetNextRows
 - C 155
 - Visual Basic 298
- TM1SQLTableNumberOfColumns 380
- TM1SQLTableNumberOfRows 380
- TM1SQLTableRowsetSize 380
 - properties 380
- TM1SubsetAlias 381
- TM1SubsetAll
 - C 155
 - elements 44
 - Visual Basic 299
- TM1SubsetCreateByExpression 156, 300
 - C 156
 - Visual Basic 300
- TM1SubsetCreateEmpty
 - C 157
 - handle 44
 - Visual Basic 301
- TM1SubsetElementDisplay 47
 - C 157
 - Visual Basic 301
- TM1SubsetElementDisplayEll
 - C 158
 - Visual Basic 302
- TM1SubsetElementDisplayLevel
 - C 159
 - Visual Basic 303
- TM1SubsetElementDisplayLine
 - C 159
 - Visual Basic 304
- TM1SubsetElementDisplayMinus
 - C 160
 - Visual Basic 304

- TM1SubsetElementDisplayPlus
 - C 161
 - Visual Basic 305
- TM1SubsetElementDisplaySelection
 - C 161
 - Visual Basic 306
- TM1SubsetElementDisplayTee
 - C 162
 - Visual Basic 306
- TM1SubsetElementDisplayWeight
 - C 162
 - Visual Basic 307
- TM1SubsetElementListByIndexGet
 - C 163
- TM1SubsetElementListByIndexGetEx
 - C 163
- TM1SubsetElementListByNamesGet
 - C 164
- TM1SubsetElements 381
- TM1SubsetExpression 381
- TM1SubsetInsertElement
 - add to empty 44
 - C 165
 - Visual Basic 307
- TM1SubsetinsertSubset
 - C 165
- TM1SubsetInsertSubset
 - Visual Basic 308
- TM1SubsetListGet
 - C 166, 167
- TM1SubsetSelectByAttribute
 - C 168
 - Visual Basic 310
- TM1SubsetSelectByIndex
 - C 169
 - Visual Basic 311
- TM1SubsetSelectByLevel
 - C 170
 - Visual Basic 311
- TM1SubsetSelectByPattern
 - C 170
 - Visual Basic 312
- TM1SubsetSelectionDelete
 - C 171
 - Visual Basic 313
- TM1SubsetSelectionInsertChildren
 - C 172
 - Visual Basic 314
- TM1SubsetSelectionInsertParents
 - C 173
 - Visual Basic 315
- TM1SubsetSelectionKeep
 - C 173
 - Visual Basic 316
- TM1SubsetSelectNone
 - C 174
 - Visual Basic 316
- TM1SubsetSort
 - C 174
 - Visual Basic 317
- TM1SubsetSortByHierarchy
 - C 175
 - Visual Basic 318
- TM1SubsetSubtract
 - C 176
 - Visual Basic 318
- TM1SubsetUpdate
 - C 176
 - overwriting subset 44
 - Visual Basic 319
- TM1SystemAdminHostGet 177
- TM1SystemAdminHostSet 3, 8, 177
 - C 177
 - tm1admserv.exe 3
 - Visual Basic 320, 321
- TM1SystemBuildNumber 178
- TM1SystemBuildNumber_VB 321
- TM1SystemClose
 - C 178
 - cube 40
 - deleting dimension 37
 - deleting subset 44
 - Visual Basic 322
- TM1SystemGetAdminSSLCertAuthority
 - C 54
- TM1SystemGetAdminSSLCertID
 - C 54
- TM1SystemGetAdminSSLCertRevList
 - C 54
- TM1SystemGetServerConfig
 - C 179
- TM1SystemOpen
 - C 180
 - Visual Basic 322
- TM1SystemProgressHookSet
 - C 180
 - callback 9
- TM1SystemServer
 - requirements 8
- TM1SystemServerClientName
 - C 181
- TM1SystemServerClientName_VB
 - Visual Basic 323
- TM1SystemServerConnect 8
 - C 182
 - Visual Basic 324
- TM1SystemServerConnectIntegratedLogin
 - C 183
 - Visual Basic 325
- TM1SystemServerDisconnect 8
 - C 183
 - Visual Basic 326
- TM1SystemServerHandle 8
 - C 184
 - Visual Basic 326
- TM1SystemServerName 8
 - C 185
- TM1SystemServerName_VB
 - Visual Basic 327
- TM1SystemServerNof 8
 - C 185
 - Visual Basic 328
- TM1SystemServerReload 8
 - C 186
- TM1SystemServerStart
 - C 186
 - Visual Basic 329
- TM1SystemServerStartEx
 - C 187
- TM1SystemServerStop
 - C 188
 - Visual Basic 329

TM1SystemSetAdminSSLCertAuthority	TM1VallsUndefined
C 53	C 196
TM1SystemSetAdminSSLCertID	Visual Basic 338
C 54	TM1VallsUpdatable
TM1SystemSetAdminSSLCertRevList	C 197
C 54	Visual Basic 339
TM1SystemSetAdminSvrExportKeyID	TM1ValObject
C 54	C 198
TM1SystemSetExportAdminSvrSSLCertFlag	handle 12
C 54	Visual Basic 339
TM1SystemSystemServerReload	TM1ValObjectCanRead
Visual Basic 328	C 198
TM1SystemVersionGet	rights 12, 16
C 189	Visual Basic 340
release 9	TM1ValObjectCanWrite
Visual Basic 330	C 199
TM1U 5	rights 12, 16
TM1UserKill	Visual Basic 340
C 189	TM1ValObjectGet
TM1ValArray	C 199
C 189	value capsule 12
TM1ValArrayGet	Visual Basic 341
C 190	TM1ValObjectSet
returns 12	C 200
Visual Basic 330, 331	value capsule 12
TM1ValArrayMaxSize	Visual Basic 342
C 191	TM1ValObjectType
Visual Basic 332	C 200
TM1ValArraySet	returns 12
C 191	Visual Basic 342
Visual Basic 332	TM1ValPoolCount
TM1ValArraySetSize	C 201
C 192	Visual Basic 343
Visual Basic 333	TM1ValPoolCreate
TM1ValArraySize	C 201
returns 12	Visual Basic 343
TM1ValBool	TM1ValPoolDestroy
C 192	C 201
Visual Basic 334	Visual Basic 344
TM1ValBoolGet	TM1ValPoolGet
C 193	C 202
retrieving call result, C 62	Visual Basic 344
retrieving call result, Visual Basic 231	TM1ValPoolMemory
Visual Basic 334	C 203
TM1ValBoolSet	Visual Basic 345
C 193	TM1ValReal
Visual Basic 335	C 203
TM1ValErrorCode	Visual Basic 345
C 194	TM1ValRealGet
values 17	C 204
Visual Basic 335	Visual Basic 346
TM1ValErrorString	TM1ValRealSet
C 194	C 204
TM1ValErrorString_VB	Visual Basic 347
Visual Basic 336	TM1ValString
TM1ValIndex	C 204
C 195	Visual Basic 347
Visual Basic 336	TM1ValStringEncrypt
TM1ValIndexGet	C 205
C 195	Visual Basic 348
Visual Basic 337	TM1ValStringGet
TM1ValIndexSet	C 206
C 196	value capsule 11
Visual Basic 338	TM1ValStringGet_VB
TM1VallsChanged	Visual Basic 234, 349
C 196	TM1ValStringMaxSize
	C 206

- TM1ValStringMaxSize *(continued)*
 - value capsule 11
 - Visual Basic 349
- TM1ValStringSet
 - C 207
 - Visual Basic 350
- TM1ValStringSetUTF8
 - C 208
- TM1ValStringSetW
 - C 208
- TM1ValStringUTF8MaxSize
 - C 209
- TM1ValStringWMaxSize
 - C 207
- TM1ValType
 - C 210
 - error code 17
 - Visual Basic 350
- TM1ValTypeEx
 - C 211
- TM1ValTypesBinary
 - C 211
- TM1ValTypesString
 - C 211
- TM1ViewArrayColumnsNof
 - C 211
 - Visual Basic 351
- TM1ViewArrayConstruct
 - C 212
 - Visual Basic 352
- TM1ViewArrayDestroy
 - C 213
 - Visual Basic 353
- TM1ViewArrayRowsNof
 - C 213
 - Visual Basic 353
- TM1ViewArrayValueByRangeGet
 - C 214
- TM1ViewArrayValueGet
 - C 214
 - Visual Basic 354
- TM1ViewArrayValuePickListByRangeGet
 - C 216
- TM1ViewArrayValuePickListExists
 - C 217
- TM1ViewArrayValuePickListGet
 - C 215
- TM1ViewCellsValueGet
 - C 218
- TM1ViewCellValueGet
 - C 217
- TM1ViewColumnSubsets 382
- TM1ViewCreate
 - C 218
 - Visual Basic 355
- TM1ViewExtractComparison 382
- TM1ViewExtractCreate
 - C 219
 - Visual Basic 356
- TM1ViewExtractDestroy
 - Visual Basic 357
 - Visual C 220
- TM1ViewExtractGetNext
 - C 220
 - Visual Basic 357
- TM1ViewExtractRealLimitA 382
- TM1ViewExtractRealLimitB 383

- TM1ViewExtractSkipConsolidatedValues 383
- TM1ViewExtractSkipRuleValues 383
- TM1ViewExtractSkipZeroes 383
- TM1ViewExtractStringLimitA 384
- TM1ViewExtractStringLimitB 384
- TM1ViewFormat 384
- TM1ViewFormatString 385
- TM1ViewPreConstruct 386
- TM1ViewRowSubsets 387
- TM1ViewShowAutomatically 387
- TM1ViewSuppressZeroes 387
- TM1ViewTitleElements 388
- TM1ViewTitleSubsets 388
- TM1ValPoolCreate
 - retrieving data 11
- Transaction Log 19
- transaction log file
 - contents 18
 - structure 19
- tree, subset element display 47
- types
 - attributes 359
 - element 370

U

- update
 - BLOB 51
 - subset 44
- updating dimension 37
- user functions
 - C 65
- user handle 5
- users
 - adding, C 62
 - adding, Visual Basic 230
 - connecting to API 180
 - connecting to API, Visual Basic 322
 - group, assigning to 231
 - group, assigning to C 62
 - group, removing from 64, 233
 - passwords, C 150
 - passwords, Visual Basic 297

V

- value array functions
 - C 192
 - Visual Basic 334
- value capsule 9
 - Boolean, C 192, 193
 - Boolean, Visual Basic 334, 335
 - constructing 11
 - constructing, C 189
 - constructing, containing index 195, 336
 - constructing, containing object handle 198, 339
 - defined 1
 - retrieving contents, C 195
 - retrieving contents, Visual Basic 337
 - string 11
 - string size, C 206
 - string size, Visual Basic 349
 - updating, index 196, 338
 - updating, string 207, 350
 - updating, TM1ValSet functions 12
- value capsule functions 53

- value pools 5, 9
 - constructing, real value 203, 345
 - constructing, string 204, 347
 - creating 11, 13
 - creating, C 201
 - creating, Visual Basic 343
 - destroying, C 201
 - destroying, Visual Basic 344
 - extracting number of values 13
 - managing 13
 - memory consumed 13
 - retrieving, contents of 204, 346
 - retrieving, memory being used 203, 345
 - retrieving, number of objects 201, 343
 - retrieving, value 202, 344
 - updating value real, C 204
 - updating value real, Visual Basic 347
- values
 - data types 10
 - error codes 11
 - extracting 11
 - functions 210, 211
 - pending 13
 - pools 9
 - retrieving, C type 210
 - retrieving, sets 13
 - retrieving, Visual Basic type 350
 - returning type 11
 - value capsules 9
- view
 - as child object 39

- view (*continued*)
 - constructing view array, C 212
 - constructing view array, Visual Basic 352
 - creating from cube, C 218
 - creating from cube, Visual Basic 355
 - destroying view array, C 213
 - destroying view array, Visual Basic 353
 - functions 50
 - retrieving, number of columns view array 211, 351
 - retrieving, number of rows view array 213, 353
 - retrieving, single value 214, 354
- Visual Basic
 - adding code to project 23
 - creating project with Microsoft Visual Studio 23
 - logging in to server 24
 - logging out from server 25

W

- weight
 - component, C 98
 - component, Visual Basic 251
 - consolidated element, C 162
 - consolidated element, Visual Basic 307
- WRITE access
 - preventing, C 131, 132
 - preventing, Visual Basic 278, 279
 - removing lock, C 132
 - removing lock, Visual Basic 279